Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Master Thesis

Implementation and Evaluation of Profile-based Prediction for Energy Consumption in a Cloud Platform

---

Author: Alejandro Olvera Anton

Advisors:
prof. Fatos Xhafa

October 2017

Acknowledgments

To prof. Fatos Xhafa for helping and guiding me throughout this project.

To the RDLab cluster team for giving me the chance of using their infrastructure.

To my family for giving me the opportunity to study at this University, where I have met great and brilliant people with whom I have shared both personal and professional experiences that I will never forget.

To my friends for their support in the most important moments of my studies.

Abstract

Cloud Computing services are essential to modern society, by making possible the technologies we use everyday. The increasing number of people and organizations using these services result to higher demand in datacenters, raising energy consumption and carbon footprint. Reducing the cost of energy consumption has become a subject of interest for many researchers, who approach the problem from various points of view, starting with different optimization processes involved, scheduling algorithms, modeling and prediction. The main purpose of this project is to give an extensive vision of the steps followed by a datacenter upon the arrival of a task, by showing how it travels along their processing timeline focusing on the energy-aware point of view of the datacenter. This representation is achieved by using a datacenter simulator to avoid high infrastructure costs. Some simulations were executed to show a comparison of the selected algorithms in two scenarios: empty and loaded datacenter. The results are evaluated in terms of energy consumption, quality of service, resource memory efficiency among others.

Some models were built to predict the energy consumption based on the simulation attributes from the simulation data. These models were evaluated and discussed in both scenarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cloud computing has become one of the current trends in the modern society for the facilities of usage and interact with our digital data. The amount of data and services have been exponentially increased over the years, and their management turned into a complex scenario. Data centers that manage cloud user requests use critical computing infrastructures, while other areas like Big Data and the Internet of Things have also incremented their demand, and the scenario predicts a higher use of these infrastructures. The proliferation of datacenters with inefficient scheduling of their resources results in high energy consumption, carbon footprint and monetary cost. Several studies of green strategies have carried out, where the implementation demonstrate to be capable of reducing energy consumption and cost by 40% [1], whereas other reviews consider an energy saving up of 75% using servers and networks strategies [2]. The resource utilization is a relevant point for the datacenter efficiency as well. Currently, there is a widespread awareness on developing energy-efficient measures that simultaneously maximizes performance efficiency and minimize energy consumption [1]. In [3] is shown an example of resource usage improvement, where the hardware utilization improved by up to 70% using the virtualization. According to [3], one datacenter can generate 170 million metric tons of CO_2 per year, where the expected carbon emissions by datacenters worldwide in 2020 is about 670 million metric tons annually. The total estimated energy for datacenters in 2013 was 91 billion kilowatt-hours of electricity. The typical datacenter consumes as much energy as

25.000 households per year. As a result of such potential impacts to the environment, the green cloud computing initiative has emerged as part of the green IT vision [4]. This article considered the best practices that concerns energy saving and productive utilization, such as turning off the computers when they are not used, keeping server monitors off, purchasing products with energy star, reusing and recycling hardware devices, and a long list of strategies to contribute with the eco-friendly environment, also known as a Green IT. The overall objective of Green IT is to increase energy efficiency and reduce CO_2 emissions to save the environment. Some important governments have taken position to confront this problem. USA invests 90 billion dollars for green jobs initiatives, which promotes the solar and wind technologies, improving the energy efficiency including home retrofits, etc. via the American Recovery and Reinvestment Act (ARRA). The United States Energy Department granted $47 million of the ARRA money towards projects that optimize datacenter software and hardware, shown in [5].

A datacenter generates a high energy consumption that needs a substantial operational, maintenance and cooling costs of data centers components. The increment of resource usage produces overrunning in the infrastructure capacity which surpasses the current energy requirements. This contributes to excessive heat discharge which leads to the reduced lifetime of IT components thereby adversely affecting the reliability of data centers. The need of green strategies has become something essential in the next decades for the datacenter daily usage.

To study this green strategies, we present some algorithms and heuristics used in the literature to reduce energy consumption while maintaining a good service to the client. This study is based principally in datacenters which use virtualization. Virtualization is a hot topic in Information Technology (IT). It is a good solution to abstract the physical structure and it is easy to administrate between several machines, because it divides logically the server resources as virtual machines (VMs). According to [2], once the physical server is partitioned, each VM runs an independent operating system and applications. This allows the distribution and exchange of virtual machines between different types of servers without limitation in the in-

frastructure, giving the portability, manageability and the ease of scalability in the datacenter. VMs provide isolation, which means that the resources of one VM are not used by another VM in the same physical machine. This isolation improves the security as discussed in [6], because it allows the consideration of some intrusion detection and prevention systems on the virtualization layer that allows the monitoring, the use of snapshots, etc.

In addition to using virtualization for the servers, application assignment to VMs is considered in the study as a point of research, where it shows how can be selected the applications in terms of different criteria. The resource requirements of the applications have been changed during the last years, considering more variability in their characteristics which make the problem more complex.

Energy consumption could be predicted based in the history usage of a datacenter. A model can be generated by selecting representative attributes, such as the number of incoming applications, the number of virtual machines, the energy-aware algorithms, the resource usage efficiency among others. We could consider various profiles that help the model to predict this data more accurately, like the scheduling algorithms that consume a similar power consumption, or the effect of assigning some applications to certain virtual machines. It shows global information about the datacenter environment that recommends the best suitable configuration for a determined scenario.

In the followings sections of this Chapter, the *background* is given a perspective of different layers that could be seen in a datacenter, while we continue to discuss the *objectives* of this project in detail. Chapters 2-6 are about the technical aspects of the project. Chapter 2 discusses the *state of the art* of this problem, and we consider Cloud Computing, Datacenters Simulators, Workflow Scheduling Techniques and Cloud Computing Datasets. In *Chapter 3: CloudSim* is discussed the datacenter simulator selected, and the large number of algorithms that are considered in the optimization energy process. *Chapter 4: Architecture* describes the infrastructure, the conceptual model and the optimization process used in the project. *Chapter 5: Experimental Study* shows the experiments that are conducted and the obtained

results are presented. *Chapter 6: Conclusions* shows the conclusions that are drawn, and the objectives accomplished in addition to future recommendations. *Chapter 7: Economic Report* describes the main tasks and their distribution across time, and the identified costs of the project.

1.1 Background

To address the challenge that this project has, a brief background of a datacenter is discussed. As it is shown in Figure 1-1 and according to [1], software energy management in data centers could be implemented in three layers: application, virtual machine and physical machines or servers. The two principal problems are showed: application placement and VM placement.

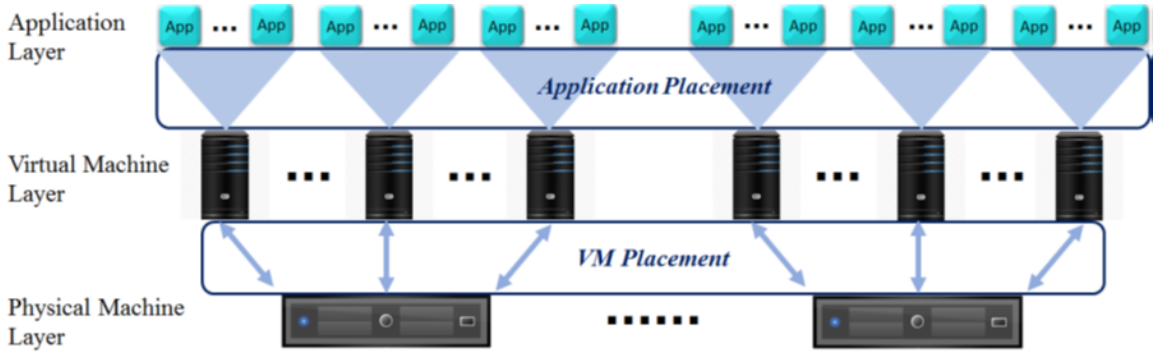

Figure 1-1: Three-layer Data Center Architecture

The responsibilities of the physical machine (PM) layer are the server resource usage, ON/OFF operations, sleep cycles, cooling and dynamic voltage frequency scaling (DVFS). DVFS is a feature of the processors that allows software to change principally their voltage and frequency in real time to achieve the best performance or lowest power possible.

VM layer is responsible for VM management, including VM placement, sizing and migration.

Application layer assigns incoming applications requested by cloud consumers to VMs for execution.

Some algorithms research the best path to solve the main problems considered, both considered NP-hard problems with a very large solution space [1]. This project shows some techniques and algorithms that could be used in the datacenter to deal with that problems, keeping in mind the trade-off about the energy consumption and the performance among others characteristics.

1.2 Objectives

The main objective of this project is to understand the process that a datacenter follows when executes different processes in a real scenario, and the algorithms associated in this flow which contribute to decrease the energy consumption while considering others essential characteristics such as the quality of service (QoS) for the client.

Keeping this goal in mind, several other objectives should be completed at the end of this project which described in Table 1.1.

Objective	Description
Main	**Implementation and Evaluation of Profile-based Prediction for Energy Consumption in a Cloud Platform**
O1	Implementation of scheduling algorithms in CloudSim
O2	Construct the dataset
O3	Analyze the dataset by different criteria
O4	Build a model to predict the energy consumption and others features based in different algorithms
O5	Self development

Table 1.1: Objectives

O1: Implementation of energy-aware algorithms in CloudSim. This objective needs to study the different alternatives that we have for scheduling. It is necessary to select the most appropriate algorithm and implement them in the simulator of the datacenter.

O2: Construct the dataset: Once we have implemented the selected algorithms, it is necessary to construct some datasets considering different parameters, like the size of the applications, the number of machines, the resource requirements, and other features used in the simulation focusing in the behavior of these algorithms.

O3: Analyze the dataset by different criteria: Datasets are created and we analyze the results to understand the environment.

O4: Build a model to predict the energy consumption and others features based in different algorithms: The creation of a model to predict energy consumption will help us to see how predictable are the datacenters scenarios.

O5: Self development: Study and learn new technologies, like the literature of the algorithms involved in this project or the datacenter simulator used.

Chapter 2

State of the art

Taking into consideration the rapid growth in cloud services that are available from different areas (education, scientific, enterprises, etc.), the evaluation of what criteria differentiate the cloud systems is a necessary step for further investigation. Cloud computing has become a current trend in the IT technologies, and society get used to it without realizing, giving an increment of processing and data associated in the datacenters. Real datacenters for research is not a viable solution by the large amount of resources and costs used, and it is necessary to find alternatives to analyze their behaviour.

In this chapter we discuss further about cloud computing and their main principal characteristics. We continue with some data center's simulators that reproduce a real scenario in a cloud environment. Some workflow scheduling techniques for cloud resources are presented, which deals with the problem of mapping tasks with resources. Finally, the most relevant cloud computing datasets used in the literature are shown.

2.1 Cloud Computing

Cloud computing is an information technology paradigm. According the National Institute of Standards and Technology (NIST), this technology has been defined as follows: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [7].

A cloud computing system is identified by some essential characteristics. These are described in the next list.

1. *On-demand self-service*: Users are able to access cloud computing provision without requiring human interaction, mostly done through a web-based self-service portal.

2. *Broad network access*: Cloud computing resources are accessible over the network and accessed through standards mechanisms, supporting heterogeneous client platforms such as mobile devices and workstations.

3. *Resource pooling*: Physical and virtual resources are dynamically assigned and reassigned according to consumer demand. The user generally has no control or knowledge over the exact location of the provided resources, only a higher level of abstraction such as the country, state or datacenter. These resources include storage, processing, memory, network bandwidth and virtual machines.

4. *Rapid elasticity*: Resources are provisioned and released on-demand or automatically based on triggers or parameters. This make sure your application will have exactly the capacity it needs at any point of time, giving to the user the possibility to get and purchase more resources in any quantity at any time.

5. *Measured Service*: Resource usage is monitored, measured and reported transparently based on utilization. This is useful to optimize resources and give to the customer clarity in their payments.

The infrastructure that it is based the development of the project is the Cloud. For this reason, some relevant topics associated to this technology are presented. The section is structured as follows:

- *Section 2.1.1*: a discussion between the differences of cloud and grid computing
- *Section 2.1.2*: presentation of the service models that a cloud system can follow
- *Section 2.1.3*: presentation of the main deployment models that exist in cloud systems

2.1.1 Cloud or Grid Computing

Cloud computing and Grid computing are two concepts that engineers confuse for their similarity in theory. In this section it is clarified what characteristics represents each of them and their main similarities and differences considering different approaches. Both concepts have been developed for the purpose of distributed computing, which means the computation of an element over a large area of machines. The information and images were obtained from [8].

Cloud Computing

Cloud refers to a range of scalable services that a user can access via an Internet connection, preferable one with a higher bandwidth and low latency. For the end users, it is just getting output for certain inputs, where the complete process that leads outputs is purely invisible. Computing is based on virtualized resources which are placed over multiple servers in clusters.

Cloud models available services are IaaS, PaaS and SaaS. These services that are in the cloud require the heavy lifting of someone else's infrastructure. Cloud computing eliminates the costs and complexity of buying, configuring and managing the hardware and software needed to build and deploy applications, where these applications are delivered as a service over the Internet.

Grid Computing

Grid computing is the collection of resources of multiple computers in a network to reach a common goal at the same time. It is focused on the ability to support computation across multiple administrative domains that sets it apart from traditional distributed computing. This comparison is shown in *Figure 2.1*, where multiple resource managers are involved in the processing, while in distributed computing scenario only uses a centralized resource manager and all the nodes cooperate together as a single unified resource or a system.

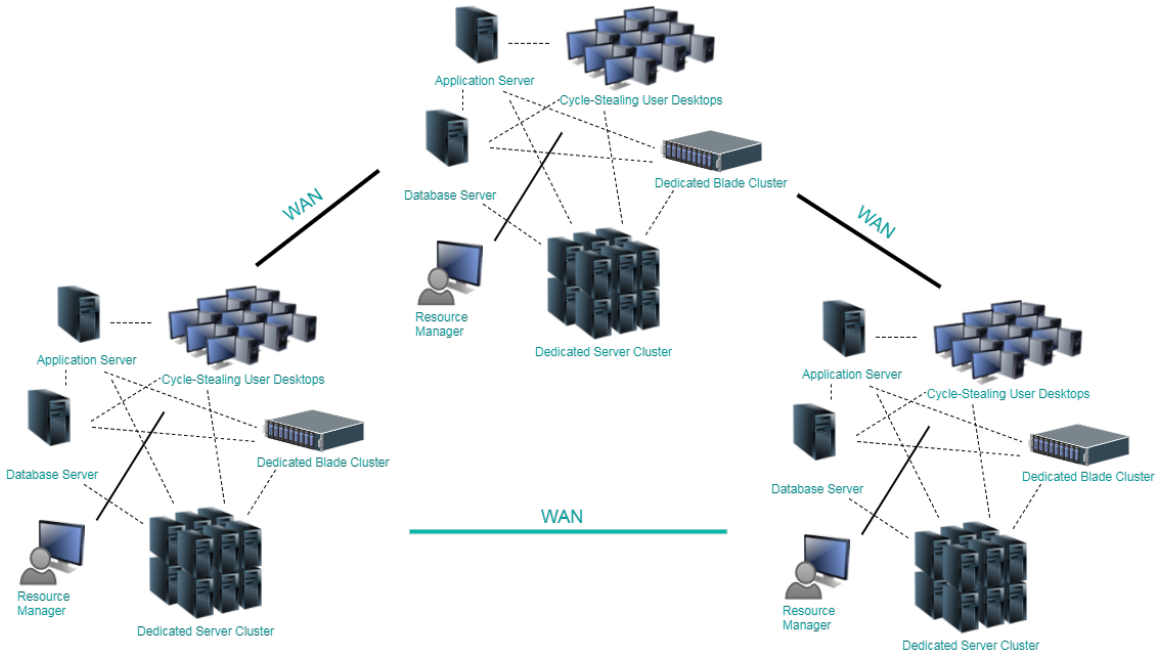

Figure 2-1: Grid Computing

Grid Computing considers the efficient utilization of a pool of heterogeneous systems with optimal workload management utilizing an enterprise's entire computational resources such as servers, networks, storage and information, acting together to create one or more large pools of computing resources. There is no limitation of users, departments or originations in grid computing.

This offers a way of using the IT resources optimally inside an organization involving virtualization of computing resources. Its concept of support for multiple administrative policies, security authentication and authorization mechanisms enables

it to be distributed over a local, metropolitan or wide-area network.

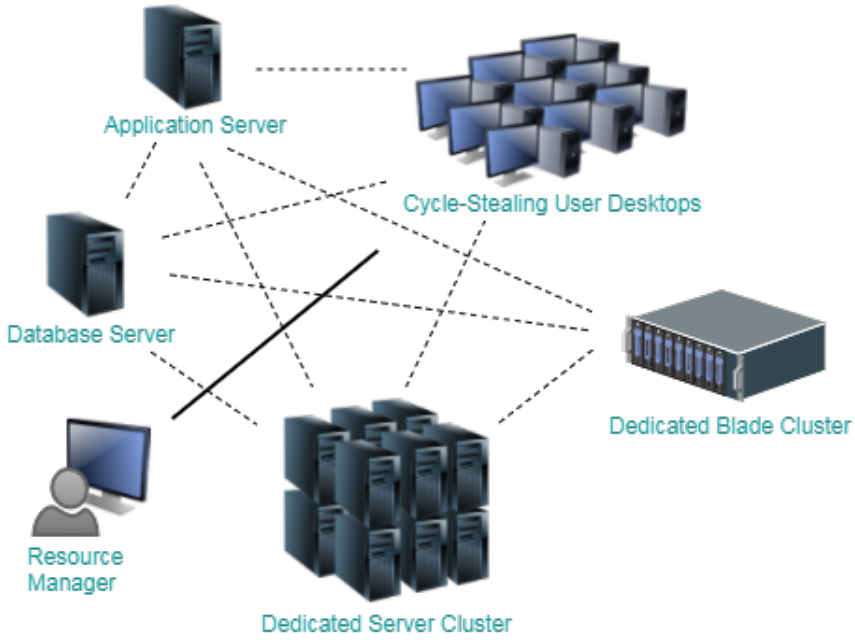

Figure 2-2: Distributed Computing

Similarities and Differences

Some similarities and differences are discussed below.

1. Cloud and Grid are scalable. Scalability is accomplished through load balancing of application instances running separately on a variety of machines on demand.

2. Both computing types involve multitenancy and multitask, meaning that many customers can perform different tasks, accessing a single or multiple application instances.

3. Cloud and Grid computing provide service-level agreements (SLAs) for guaranteed uptime availability.

4. In Grid, the infrastructure is owned by the participants, while in Cloud is provided by third parties which means that there is a client-provider relationship.

5. Cloud computing uses majority virtualization unlike Grid that it is not a commodity.

Generally, cloud is considered as the services that a customer can use in the infrastructure, while the grid is the path to optimize the computation of multiple data using techniques of distributed computing. For this reason, a cloud would usually use a grid, but a grid is not necessarily a cloud or part of the cloud.

2.1.2 Service Models

Cloud computing providers offer their services according different models. The three standard models per National Institute of Standards and Technology (NIST) are: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Each of them present different privileges and range of functionalities that the user could use in the cloud. These models are shown in detail below, where Container as a Service (CaaS) is also considered by the increase of demand in the cloud.

Infrastructure as a Service

Infrastructure as a Service (IaaS) is the model which provides to the user a high control of the resources in the cloud. The capability provided to the consumer is to provision processing, storage, networks and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems, virtual machines and applications. The consumer does not manage or control the underlying cloud infrastructure, but it has the control of the operating systems, storage, and deployed applications possibly limited for the control of some networking components (e.g., host, firewalls).

The cost is based on the resources and services used (time, bandwidth, transactions, storage, etc.), known as an Operating Expense (OpEx) category or as pay-as-you-go cost model. The payment is generally made every prefixed time (monthly, yearly, etc.).

Some of the most common IaaS solutions are Amazon Web Services (AWS) and Google Cloud Platform (GCP).

Platform as a Service

Platform as a Service (PaaS) is the category that provides a platform which allows users to develop, run and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. It provides a computing platform and a solution stack as a service with tools and libraries for the customer. Examples of this solution are based on Software Development Kits (SDKs) and Application Programming Interfaces (APIs) which allow a good and easy development environment for applications.

Some PaaS providers charge a flat monthly or yearly fee to access their service, as well as the apps hosted within it.

Some of most common PaaS solutions are Google App Engine, Amazon Elastic Beanstalk, Cloud Foundry, AppScale or Heroku.

Software as a Service

Software as a Service (SaaS) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet. SaaS removes the need for organizations to install and run applications on their own computers or in their own data centers. This eliminates the expense of hardware acquisition, provisioning and maintenance, as well as software licensing, installation and support.

The customers are subscribed to a SaaS offering rather than purchasing software to install, which generally are paid every prefixed time (monthly, yearly, etc.) using a pay-as-you-go model. Customers can rely on a SaaS provider to automatically perform updates and patch management.

The main disadvantages are that customers must rely on outside vendors to provide the software subject to the rules that the SaaS has in terms of performance or QoS. This should be agreed and measured using the Service Level Agreement ((SLA)).

Some software examples that use this model are SalesForce, Microsoft Office 365, Google Apps, AWS, Concur, Dropbox, Slack, and a wide list of well-known applications used daily.

Container as a Service

Containers as a Service (CaaS) is a form of container-based virtualization in which container engines, orchestration and the underlying compute resources are delivered to users as a service from a cloud provider.

A container is the element used in this model, which is an OS-level virtualization method for deploying and running distributed applications without launching an entire VM for each application. These containers are multiple isolated systems that run on a single control host and share the same kernel, unlike VMs which require separate OS instances. This container hold the components necessary to run the software, such as files, environment variables or libraries, and it is sufficiently independent to constrain the access to physical resources, such as CPU and memory to not consume all the host physical resources.

CaaS permits the users upload, organize, run, scale, manage and stop containers using a provider's API calls or web portal interface. In the same way as others cloud services, users pay for the CaaS resources that they use.

CaaS gained prominence and popularity with the open source Docker. Cloud providers like Google, AWS, IBM among others offer CaaS services. For example, AWS has its Amazon EC2 Container Service (ECS) which uses the Docker container managing Amazon EC2 instances.

Comparison between Models

The standard cloud models commented in previous sections have different responsibilities, and each of them should be selected depending on the user needs. The interested clients should consider the trade-off between *Effort to Manage* and *Level to Control*, where IaaS offers a major control of the infrastructure, but it is necessary a higher effort to manage the whole cloud, whereas SaaS does not give any control to the client of the cloud system which naturally gives a lower effort to manage this cloud.

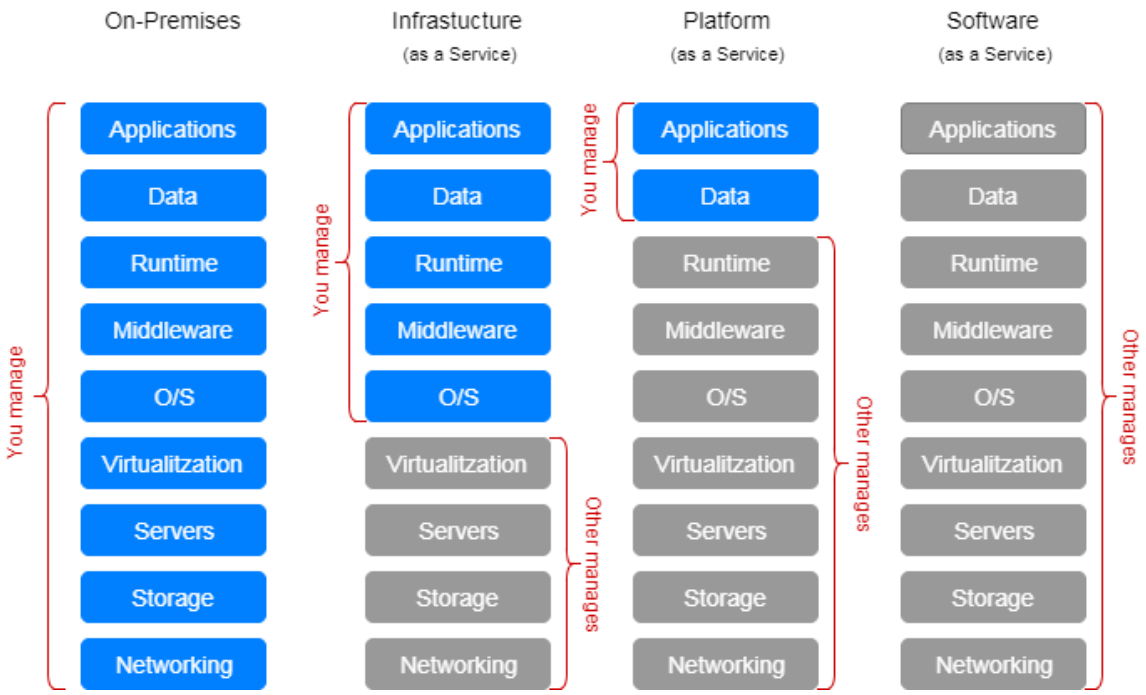

Figure 2-3: Standard Cloud Computing Models

We can say that IaaS gives users automated and scalable environments with extreme flexibility and control, while PaaS provides a framework for quickly developing and deploying applications by automating infrastructure provisioning and management. Eliminating the need to install and run programs on individual devices, SaaS makes applications available through the internet.

Besides of the three standard models, we should consider CaaS as a significant cloud model. CaaS is considered as a cloud model between IaaS and PaaS, but sometimes it is also known as a subset of IaaS. The reason is that we need to make a difference between a VM used in the IaaS or a container used in the CaaS, considering that a container can run within a VM. Also, it is possible to develop and run any application in CaaS in the same way that it is done in PaaS.

2.1.3 Deployment Models

NIST presents four deployment models as the standards [7] in a cloud system. Deployment is considered as the process of making software available and ready for use. In this section is commented the four standards models and what characteristics define each of them. Some information and images were obtained from [9].

Public Cloud

Public cloud is considered when the services are rendered over a network that is open for public use, which this cloud environment is owned by a third-party cloud provider. This cloud provider may be a business, academic, or government organization, or some combination of them, and they are the responsible for the creation and on-going maintenance of the public cloud and its IT resources.

Technically, there is little or no difference between public and private cloud architecture. However, security consideration may be substantially different for services (applications, storage, and other resources) that are made available by a service provider for a public audience, and when communication is effected over a non-trusted network.

This is the most common and well-known deployment model, where it offers the same services to all its users. The services are accessible for everyone and it is much used for the consumer segment. Examples of public services are Facebook, Google and LinkedIn. For consumers, Public Cloud offering are usually free of charge, but for professionals there is usually a pricing model.

Figure 2-4: Public Cloud Services

Private Cloud

The private cloud infrastructure is provisioned for exclusive use by a single organization used by multiple consumers. This type of clouds enable an organization to use cloud computing technology as a means of centralizing access to IT resources by different parts, locations or departments of the organization.

With a private cloud, the same organization is technically both the cloud consumer and cloud provider, where the departments requiring access to the private cloud assume the cloud consumer role.

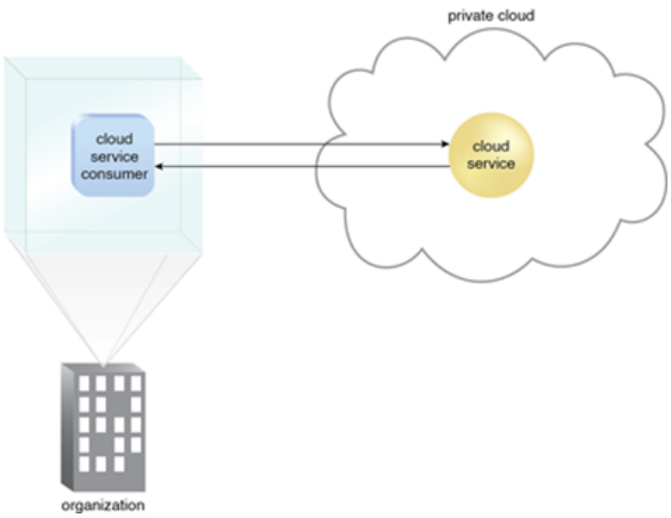

Figure 2-5: Private Cloud Infrastructure

Community Cloud

The community cloud infrastructure is similar to a public cloud, but the difference is that they are provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns, such as mission, security requirements, policy or compliance considerations. This cloud might be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them.

The member cloud consumers of the community typically share the responsibility for defining and evolving the community cloud, which it means that the community do not grant access to parties outside the community until they are accepted.

One example of this cloud could be the healthcare, where it is necessary to follow a strict standards for health-related data protection. An example is shown in Figure 2-6, where organizations access to some resources of this cloud.

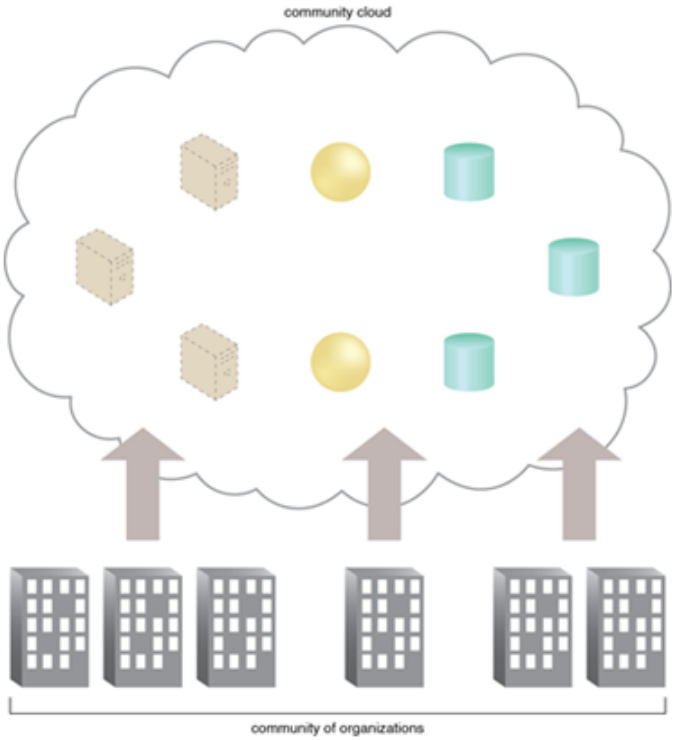

Figure 2-6: Community Cloud Services

Hybrid Cloud

The hybrid cloud infrastructure is a composition of two or more distinct cloud infrastructures, such as private, community, or public that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability. For example, a cloud consumer may choose to deploy cloud services processing sensitive data to a private cloud, and less sensitive cloud services to a public cloud. The result of this combination is a hybrid deployment model.

Hybrid deployment architectures can be complex and challenging to create and maintain due to the potential disparity in cloud environments and the fact that management responsibilities are typically split between the private cloud provider organization and the public cloud provider. An example is shown in Figure 2-7, where an organization uses a hybrid cloud architecture that utilizes both a private and public cloud.

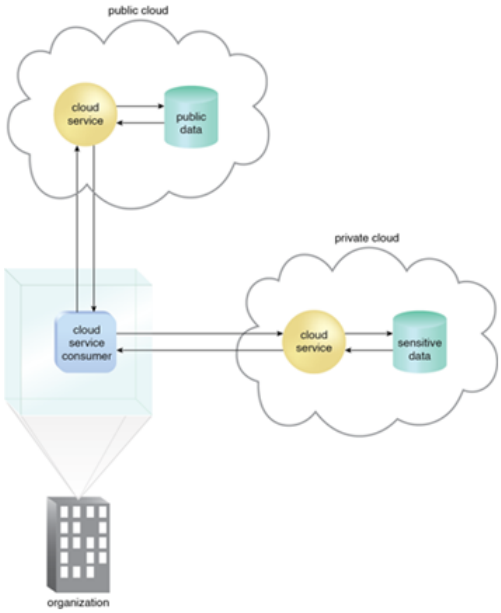

Figure 2-7: Hybrid Cloud Architecture

2.2 Datacenters Simulators

The existence of tools dedicated to simulate the environment of executions produced in a datacenter has become something necessary due the difficulty of use a dedicated scenario for testing purposes. Sometimes real testbeds like Amazon EC2 or Windows Azure cannot be enough for making the experiments since they limit the simulations in terms of scaling or monetary cost.

In this section is shown some simulation softwares discussed in [10] that are currently available on websites, introducing their functionalities and the areas that cover.

CloudSim

CloudSim is a framework that has become one of the most popular open source cloud simulators in the research and academia. It is based on the SimJava [11] which is a discrete event simulation engine, and it uses any event that occurs at a particular instant in time to modify the state in the system. It has been shown to be able to instantiate 100.000 machines in less than 5 minutes, requiring only 75MB of RAM, and it is fully written in Java.

CloudSim is a generalist simulator, which it means that it does not have any measure that it focus. It can be used for security, power consumption, containers, network topologies, federated clouds among others typical topics. The idea is to simplify the low level details related to Cloud-based infrastructures and services. However, it does not provide a GUI natively.

Due the facilities of changing the behavior of the system, some extensions of CloudSim have been presented for determined areas. *TeachCloud* was designed for educational purposes, which provides a simple graphical interface where students can modify the cloud configuration and perform simple experiments. *NetworkCloudSim* focused on the network flow model for data centers and topologies, where bandwidth sharing and latencies are involved.

Several contributions have been done in the form of projects in CloudSim, as can

be seen in [12], such as *iFogSim*, *CloudSimEx*, *CloudAnalyst* among others. Each of them provides new functionalities, such as the use of a graphical interface to configure the datacenters, or other functionalities dedicated to IoT.

GreenCloud

GreenCloud is a simulator which focuses especially on the measurement of energy consumption by the data centre IT equipment, such as computing servers, network switches and communication links [13]. It is an interesting option for the user if it is also interested in networks and routing besides the energy consumption, which gives a complete list of communication models and the support of TCP/IP.

It is released under the General Public License Agreement, and it is an extension of another tool called NS2 network simulator, which is another discrete-event network simulator used primarily in research and teaching [14].

It allows the configuration of different workload arrival rates and patterns, and can implement different power management techniques of putting components to sleep. However, it is unclear if it can be used to conduct experiments for evaluating the trade-off between performance and energy consumption in a precise manner as shown in [10], where could have some limitations in its internal design with the servers and resource management.

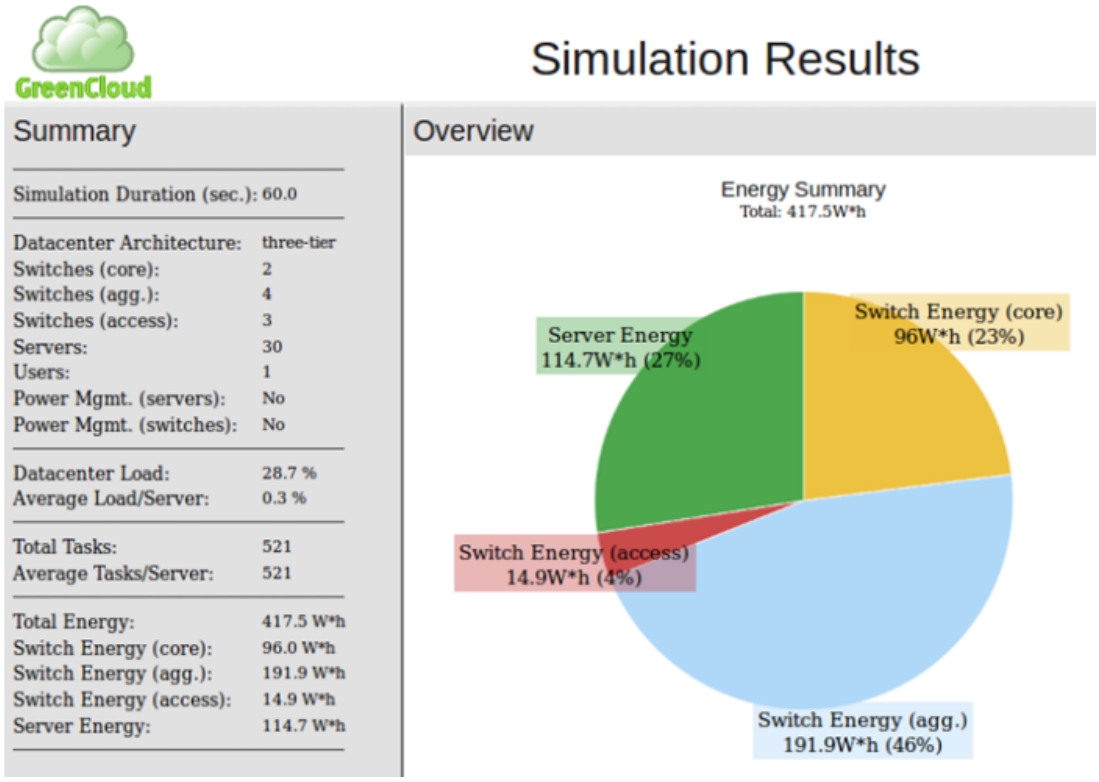

Figure 2-8: Green Cloud GUI

It is fully written in C++, and it was designed in the University of Luxembourg. It has also a graphical interface, where an overview can be seen in Figure 2-8.

iCanCloud

iCanCloud is a simulation platform for large storage networks [15]. The main objective of this tool is to predict the trade-offs between cost and performance of a given set of applications executed in a specific hardware, and then provide this information about such costs to the user. It permits a low-level detail of the simulation, and it also supports parallel execution by design.

This simulator is developed over SIMCAN, which is a simulation tool to analyze high performance I/O architectures, giving high importance to performance and QoS. For this reason, although the official website of iCanCloud considers the power consumption support in the tool, it was not created by this purpose, and some articles as [10] considered that it needs a maturity process of this area.

It has a full graphical user interface where the experiments can be designed and run. It is fully written in Java. One example of the GUI is shown in Figure 2-9, where switches, storages and racks are associated in one datacenter.

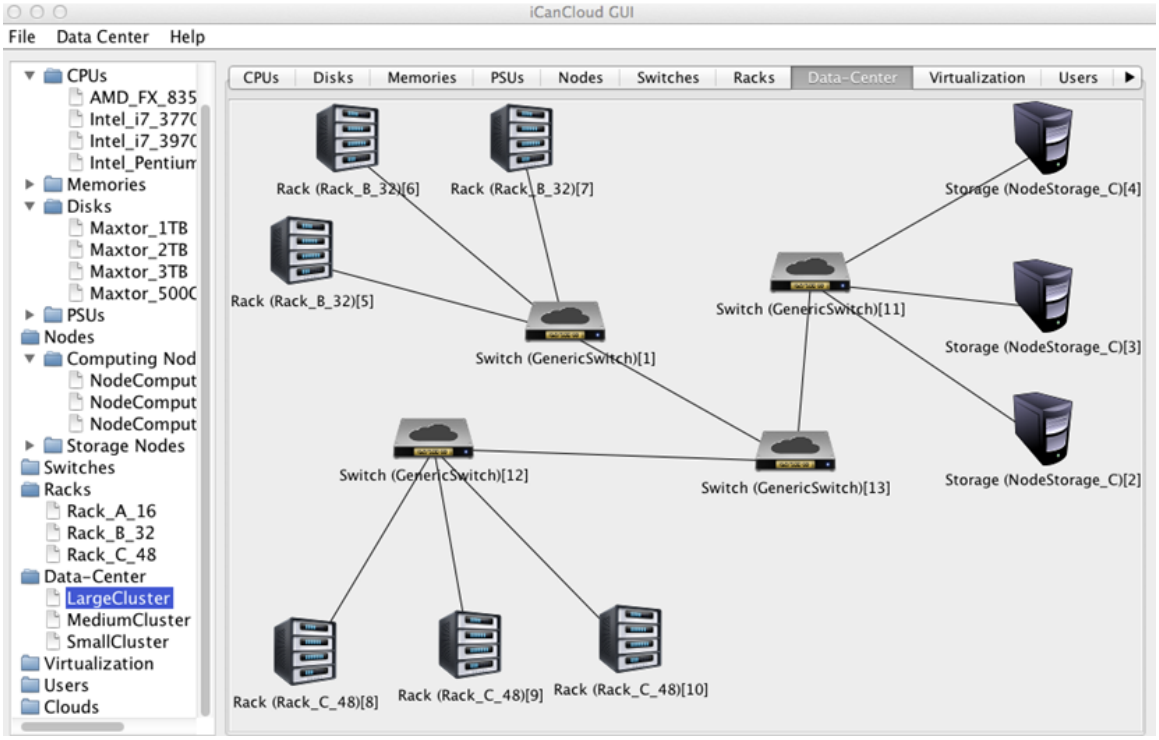

Figure 2-9: iCanCloud GUI

2.3 Workflow Scheduling Techniques

Workflow scheduling is used for the efficient execution of the workflow and utilize the cloud resources in an efficient way. It defines the placement policies that can be used in the main problems of this project, such as the application assignment used in the project, where some tasks should be placed to different VMs. A scheduling algorithm consists of three phases: (1) deciding the task mapping order, (2) scheduling task to resources and (3) task submission.

According to [16] [17] [18] [19] [20], workflow scheduling algorithm can be broadly classified into the categories shown below.

List Based Heuristics

A list based scheduling prioritizes the tasks of a workflow and schedule these tasks based on their priorities. In this approach, there are two phases: *Task Prioritizing* and *Resource Selection*. In task prioritization phase, priority is assigned to each task using a rank function, and tasks are sorted according to priority. In resource selection phase, tasks are mapped to optimal resource on the basis of priority.

Genetic Based Scheduling

Genetic algorithms (GA) provide robust search technique that allows high quality of solution to be derived from large search space in polynomial time by using principle of evolution. A number of workflow scheduling algorithms use genetic algorithms to provide robust and good solutions. There are four phases in a genetic based algorithm: selection, crossover, mutation and fitness evaluation. In this, we have to define a fitness function based upon user optimization requirement. Genetic algorithm works on the principle of survival of fittest this giving result having highest fitness value i.e. good solutions according criteria.

Ant Colony Optimization

Ant Colony Optimization (ACO) is a meta-heuristic approach which uses foraging behavior of ants. It works by simulating the pheromone-depositing and following nature of ants and has been used in various optimization problems. The main advantage of ACO for workflow scheduling is that ACO make full use of instance based heuristic information.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a self-adaptive global search based optimization technique. The algorithm is similar to other population-based algorithms like genetic algorithms, but there is no direct re-combination of individuals of the population. PSO uses social behavior of the particles. For each generation, every particle adjusts its trajectory on the basis of its best position (pbest) and position of

the best particle (gbest) of entire population. This approach increases the stochastic nature of the particle and converges quickly to global minima with a reasonable good solution. Experimental results shows the PSO based workflow scheduling algorithms give better results as compare to Genetic and ACO based solutions.

2.4 Cloud Computing Datasets

This section shows some datasets used in the literature which are used to simulate realistic data in a cloud environment where VMs are involved. This is especially useful for the dynamic VM data, because real VM resource consumption is known to be oscillating in highly non-trivial ways.

Few cloud or grid workload traces are publicly available due to privacy concerns and business competition. It is very common to find the data obfuscated applying different techniques, such as the use of a random hash in the fields, or a linear and scaling transformation of these variables.

We explain three datasets which contain a wide variety of information of the cloud environment, especially in terms of resource usage.

2.4.1 Planetlab

PlanetLab is a global research network that supports the development of network services [21]. It consists more than 1300 physical nodes worldwide that run several co-located but isolated user tasks using virtualization.

Some PlanetLab workload traces containing exclusively the CPU usage data have been used in some experiments of CloudSim [22], and they are available in the own framework. This data has been collected from 10 days and about 1000VMs using different internal machines in 2011. These data are stored in simple text files in which each line contains a single number, that it's the CPU load in the given sample as a percentage of the request capacity. However, this dataset only has this value, and not more information could be obtained, such as the PM data.

The data was collected in the CoMon project, which was a monitoring system

developed to gather and process data from PlanetLab nodes [23]. The server of the CoMon project collected a lot of useful information from about 600 nodes for several years. Unfortunately, the server broke down, also effecting the end of the CoMon project, but given this data to the CloudSim community.

2.4.2 Bitbrains

Bitbrains is a Dutch service provider specialized in managed hosting for enterprise customers that released a dataset containing workload data for altogether 1750 VMs from its hosting center, representing business-critical enterprise applications [24].

The dataset are divided by several files, where each file contains the performance metrics of a VM. These files are divided by two types of traces: FastStorage and Rnd. FastStorage consists of 1.250 VMs that are connected to fast storage area network (SAN) storage devices. The second trace, Rnd, consists of 500VMs that are connected either to the fast SAN devices or too much slower Network Attached Storage (NAS) devices. The FastStorage trace includes a higher fraction of application servers and compute nodes than the Rnd trace, which is due to the higher performance of the storage attached to the FastStorage machines. Conversely, for the Rnd trace we observe a higher fraction of management machines, which only require storage with lower performance and less frequent access. In the Rnd directory, the metrics are recorded in three different months.

For each VM file it is mainly described the VM's dynamic data, sampled every 5 minutes. These data include the CPU and memory used of the VM at the given point of time, as well as disk and network I/O throughput values. However, in this dataset we do not have information about the power management system data.

2.4.3 Google Cluster Data

In 2011, Google made a dataset publicly available which contains a resource and workload from a cluster of roughly 12.000 PMs in a period of 29 days [25]. For data protection reasons, the published dataset contains only relative numbers and all strings are obfuscated.

Google cluster is considered as a set of machines, packed into racks and connected by a high-bandwidth cluster network. These machines are grouped by cells that share a common cluster-management system that allocated work to the machines. The tasks are compressed in jobs, where each job consist of one or more tasks, each of which is accompanied by a set of resource requirements used for scheduling the tasks onto machines. There is no evidence that the tasks in the Google dataset are actually VMs. In fact, it seems that the tasks are not VMs since the majority of tasks have tiny resource consumption in the trace as discussed in [26]. For each task, we can get relevant parameters such as the requested CPU, memory and disk capacity, or the actual CPU, memory and disk size used, where some of them were regularly measured with some sample period (in most cases, the sample period was 5 minutes, but there are some deviations). The sample period resource load was measured every second.

In this dataset, we can obtain the information of users, machines, jobs and tasks among different events which are used in machines and jobs to manage their scheduling work. In general, it is a very complete dataset according the huge amount of data obtained and the type of data recollected, but some information is declared as missing from the real scenario, such as the jobs failed by dependency constraints that are not presented, or the maximum number of tasks within a job that can be on the same rack, fully explained in its schema document obtained in [27].

Chapter 3

CloudSim

CloudSim is described by Calheris et al. [2011] as a modelling, simulation and experimentation toolkit that can be used to model clouds, data centers and VMs. Technically, CloudSim is an open source library that was assembled using the Java programming language, and it was built at the University of Melbourne by the Computer Science and Software Engineering Department in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory [12]. It allows the implementation of resource allocation policies which can be studied, tested and modified. The two main advantages of CloudSim discussed in [28] are: (1) the time effectiveness, it requires little time to setup cloud environments to test applications performance; (2) the flexibility and applicability, researchers can test their tests for different cloud deployment models like public, private or hybrid with little extra overhead.

This section explains different aspects associated to the CloudSim, and the algorithms used for scheduling, taking care the goals of the project. In *architecture*, it is showed the internal architecture that CloudSim uses. *Policies* gives an introduction of the different types of policies that CloudSim allows in the applications and virtual machines. This comparison is interesting to understand how these entities work in a CloudSim scenario when they have been processed. We should consider that all these policies are not available for the power simulations.

System model and performance metrics explain how the CloudSim system works in terms of interesting areas, like the cost of the migration of a VM, and how to measure the QoS. Finally, the *algorithms* considered in the optimization process of energy consumption have been described, showing static and dynamic responsibilities assumed in the project.

3.1 Architecture

CloudSim follows a three-layered architecture shown in *Figure 3-1* from [28], consisting the simulation engine, cloud services and the source or user code. The top layer in the simulation stack is the User Code, that exposes configuration related functionalities for hosts (e.g. number of machines or their specification), the applications (e.g. number of tasks and their requirements) and their corresponding types, the VMs or the broker scheduling policies. The user can generate a mix of different configurations and scenarios at this layer and perform robust tests. The medium layer is the CloudSim layer, which provides functions for modelling virtualized Cloud-based data center. An example of functionality could be the allocation of existing VMs to hosts, besides the allocation of CPU, memory storage and bandwidth resources. It controls, during the simulation, such components as virtual machines, hosts, the applications executed, and the global datacenter. Last layer is the CloudSim Core, which is located at the bottom of the architecture. This engine is responsible for processing different simulation events and creating main entities of the cloud, such as the data center, the hosts, the VMs and the broker. It also manages the queues and handles the communication between existing entities.

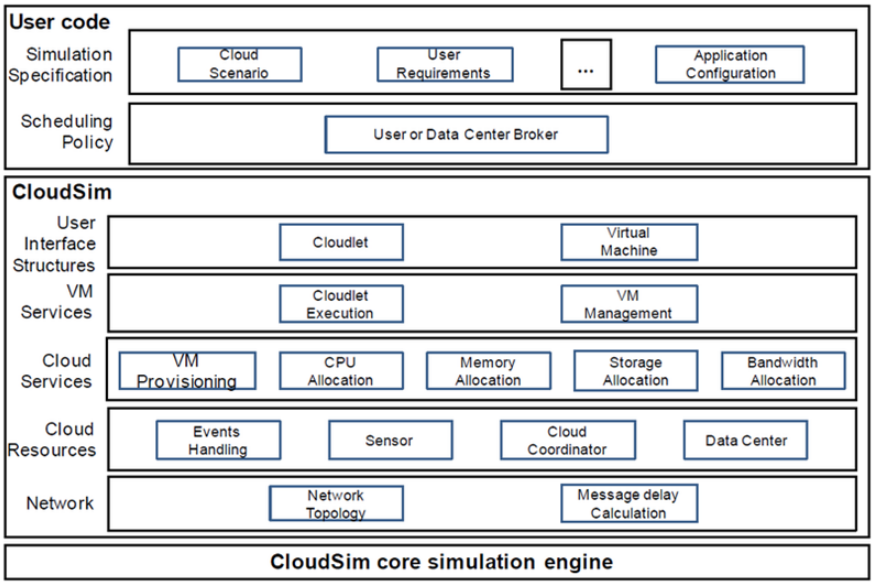

Figure 3-1: CloudSim Layered Architecture.

3.2 System Model and Performance Metrics

Given the importance to understand how CloudSim works in the system, several aspects have been recovered from [22], where Rajkumar et al. explains aspects of the system model, and how they calculate the metrics that are reutilized in the project.

This section explains in detail the following aspects: (1) the CPU architecture in the system, to give a simple understanding of what can be parallelized; (2) the power model of CloudSim, and the aspects that use the tool for obtaining the total energy generated in the simulation; (3) the method that uses CloudSim to manage the migrations of VMs, critically in the QoS in the datacenters; (4) how the QoS is calculated during the simulation.

3.2.1 CPU Architecture

CloudSim allows the use of physical servers with multi-cores, with the particularity that each core have the same capacity of processing defined in MIPS. Considering appropriate policies, the applications can be executed in parallel in the same VM, where a VM can use different virtual CPUs of different processors. Each physical CPU core has several virtual CPU's, which are simulated, and they share the hardware of

one core. However, the CPU capacity required for a VM must be less or equal to the capacity of a single physical CPU core. The reason is that, if the CPU capacity required for a virtual CPU core is higher than the capacity of a single physical core, then a VM must be executed on more than one physical core in parallel. However, automatic parallelization of VMs with a single virtual CPU cannot be assumed.

3.2.2 Power Consumption

The power consumption by servers in datacenters is mostly determined by the CPU, memory, disk storage, power supplies and cooling systems. In [22] is argued that the power consumption by servers can be described as a linear relationship between the power consumption and CPU utilization, even when Dynamic Voltage and Frequency Scaling (DVFS) is applied. It is considered that the reason lies that the CPU behavior can be modified (e.g. changing frequency or voltage) while other components such as memory and network interfaces cannot be considered.

However, the complexity of the environments give difficulties to assure this assumption, and some power models of the hosts have been considered to deal with them. Generally, we can find in CloudSim specifications of different machines that consider different loading (i.e. CPU used in the total machine) usage and, based that, it consumes a determined power. An example is shown in Table 3.1, where the server HP ProLiant G4 consumes, for instance, 108W when it arrives to 70% of CPU used. There are other models that work different. For example, given a maximum power of the machine, the power usage can be calculated by the linear or cubic calculations considering the utilization of this resource usage. CloudSim basically uses the CPU usage of the servers to calculate the total power consumption used in the simulation, because if you do not consider this usage, no energy has been consumed in the simulation, which ignores the power used by low-level elements like switches or others.

	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP Pro-Liant G4	86	89	92.6	96	99.5	102	106	108	112	114	117
HP Pro-Liant G5	93.5	97	101	105	110	116	121	125	129	133	135

Table 3.1: Example of a Power Consumption Model of a host.

3.2.3 Cost of VM Live Migration

The live migration of VMs is the capability of transferring a VM between machines without suspension and with a short downtime. However, it has a negative impact on the performance for the applications running inside this VM. The studies did by Beloglazov et al. [22] showed that these migrations produce a performance degradation that can be estimated as approximately 10% of the CPU utilization for the VM migrated, which means the utilization would be 90% in the virtual machine considering the requested MIPS used. CloudSim also applies the 10% of MIPS requested of the migrating VM into the destination host, which may cause some SLA violations. The length of a live migration depends on the total amount of memory used by the VM and the available network bandwidth. The bandwidth that a host uses is the half of its total bandwidth (the other half is for managing the bandwidth requirements of the VMs). The VM migrations follow a FIFO queue i.e. one by one are migrated, and it is not considered multiple VM migrations. CloudSim considers the images and data of VMs are stored on a Network Attached Storage (NAS), which is not used for the migration, and this information can be accessed from any host of the datacenter. This is an interesting consideration since the increment of heavy images and data make difficult or almost impossible the VM migrations.

3.2.4 Quality of Service

Quality of Service (QoS) is the description or measurement of the overall performance of a service, such can be the datacenter. Particularly, this performance is important by the users of the network. To measure the QoS, several related aspects of the network service are often considered, such as error rates, bit rate, throughput, transmission delay, availability, jitter, etc.

QoS requirements are commonly formalized in the form of SLA, which is the contractual agreement between customers and providers. This specifies an agreement for the ability of a network or protocol to give guaranteed performance, throughput or latency bounds based on a mutually agreed measures. As these characteristics can vary for different applications, it is necessary to define a workload independent metric that can be used to evaluate the SLA delivered of any VM deployed in the system. Beloglazov et al. [22] defined two workload independent metrics that can be used to evaluate the SLA delivered of any VM under IaaS. The first was the percentage of time that active hosts have experienced CPU utilization of 100% called SLATAH, being a potential host to be overloaded. The second is the degradation of performance based by VM migration (PDM) commented in the previous section. The higher these values are, the less performance and high violations of SLA will appear. Both have been considered to have a similar importance (i.e. weight) for the SLA, so they are calculated as the multiplication of both values.

3.3 Policies

CloudSim works with policies to manage the behavior of the applications and VMs when they are processed (i.e. they manage the different types of elements that CloudSim uses). Task scheduling use three types: *SpaceShared*, *TimeShared* and *Dynamic Workload*. Application space policy considers that only one cloudlet is executed simultaneously per VM, while time policy allows the execution of several applications in the VM at the same time. *Dynamic Workload* is a scheduling policy created exclusively for the power package of CloudSim, which simulates the loading

instead of a current cloudlet, adapting all the features that power package implements. This policy considers that there is just one application for the VM using this scheduler (i.e. the application will not compete for CPU with other ones).

Another well-known policies not implemented in the packages are the preemptive and non-preemptive tasks discussed in [29]. Preemptive scheduling considers the tasks that could be interrupted during the execution when an event appears, such as the incoming of higher prioritary tasks. This scheduling could stop tasks that uses a high demand of CPU for a long time. In a IaaS infrastructure results of removing a selected task in the assigned VM that it is being executed. However, it produces overheads to schedule these tasks. Non-preemtive scheduling will be the contrary of the previous scheduling (i.e. once the tasks are assigned in one VM, they will not be interrupted and moved to other VMs).

In a similar way in the VM policies, we can find the *SpaceShared* policy that allocates some CPUs to the VM that cannot be shared with others. If there aren't enough CPUs as required by a VM, or whether the available CPUs does not have enough capacity, the allocation fails. *TimeShared* in VM policies allows sharing PEs, and it considers the 10% performance degradation explained in *system model and performance*. However, if the MIPS requested for the VM is higher than the available MIPS in the host, the allocation fails. The last policy is *TimeSharedOverSubscription*, which uses the same characteristics than *TimeShared* but considering over-subscription, which means that allows the allocation of VMs that require more CPU than the VMs have, resulting in performance degradation. This is achieved by redistributing the MIPS for all the VMs of the host using a scaling factor that permits the allocation of the new VM.

A known comparison between time and space policies is shown in some official online courses of CloudSim from [30]. Figure 3-2 is the comparison of all of them that work in two axis, cores and time. For example, we can see in VM Space and Task Space that tasks t1 and t2 are executed in each core, but they do not share the CPU with others tasks until they have finished. The contrary can be seen in VM Time and Task Time, where all the tasks share the CPUs from the beginning.

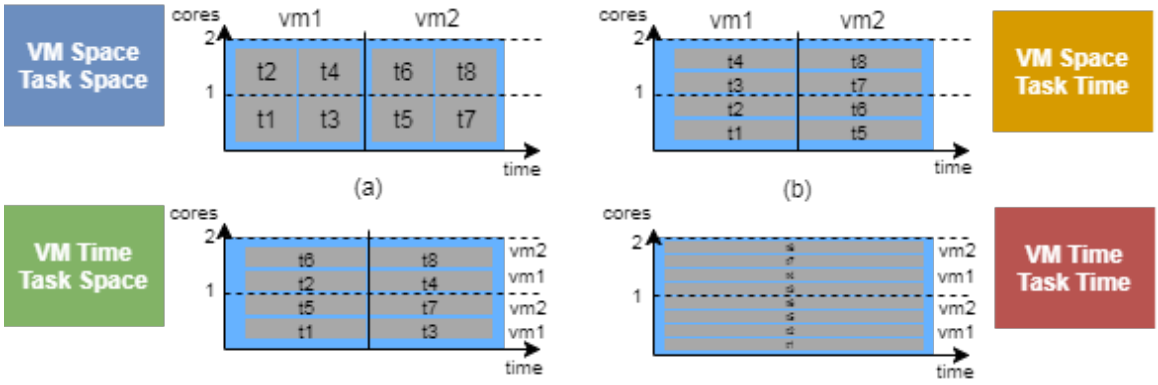

Figure 3-2: CloudSim comparison of Time and Space policies.

3.4 Algorithms

This section explains some algorithms that deal with the Task Placement and VM Placement introduced. It describes what criteria uses CloudSim to manage the migrations used in the *optimitzation process*. Static and dynamic assignments are the criteria to divide the subsections, which means the blocks that have been executed once or various times respectively.

3.4.1 Static

3.4.1.1 Task Placement

Task placement is the first assignment process that CloudSim produces in a simulation. This assignment can be seen as a bin packing problem with variable bin sizes and prices: items are the tasks that have to be allocated; bin sizes are the available CPU and memory capacities of the VMs; and prices correspond to the estimation energy cost obtained in executing the tasks allocated in the VMs considered to be used. Bin packing is a NP-hard problem, so it is not possible to find an optimal result in a reasonable time, but different approaches have been considered to deal with that problem.

Two algorithms have been considered in this step: (1) CloudSim Algorithm, the default process which does not have any optimization and it makes a trivial assignation; (2) Repairing Genetic Algorithm (RGA), an algorithm designed by Meera Vasudevan [1] which has been demonstrated that improves with a 23% less energy consumption and 43% more resource utilization in comparison with some Genetics Algorithms. Both scenarios consider that all the VMs start from an empty state.

CloudSim default assignation

This is the default assignation that CloudSim uses in the simulation package. First, it assumes that the number of applications will be the same than VMs, and it assigns each application to one virtual machine. The first cloudlet goes to the first VM, the second cloudlet with the second VM, and so on. If there are not enough VMs for the cloudlets, the execution of the cloudlet is postponed. However, the execution of cloudlet postponed is not implemented, it is necessary to implement this logic in the broker if it is desired.

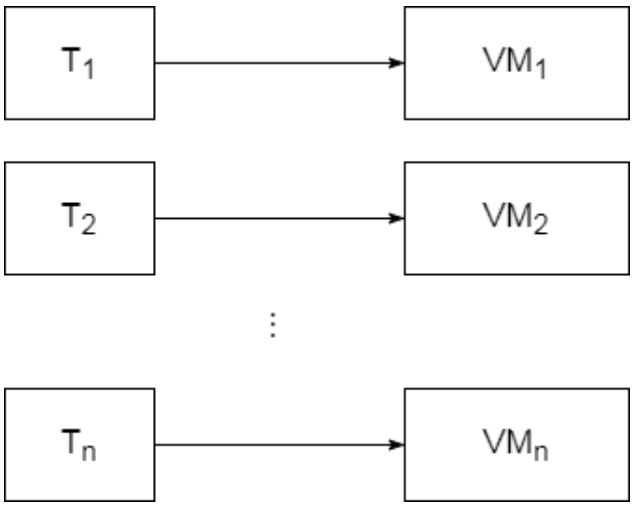

Figure 3-3: CloudSim Task Default assignation Algorithm.

Repairing Genetic Algorithm

Repairing Genetic Algorithm (RGA) is the proposed genetic algorithm designed in [1], where it considers a complete research of the application assignment, where we have only focused in the static assignment problem formulation.

This thesis shows that genetic algorithms (GA) gives better results than greedy algorithms. However, it explains the need of improvement of the classic genetic algorithm since these algorithms are imperfect, slow or no convergence. RGA tries to minimize the energy consumption and makespan whilst maximizing resource utilization. It incorporates two relevant parts: (1) Longest Cloudlet Faster Processor (LCFP), which generates an initial population; (2) Infeasible-Solution Repairing Procedure (IRP), which solves the chromosomes or solutions which are infeasible.

```
Repairing Genetic Algorithm
1  Find output of solutions generated by LCFP;
2  Initialize population with LCFP output;
3  Evaluate fitness of each candidate chromosome;
4  while Termination condition is not satisfied do
5      for Each Generation do
6          for Each chromosome do
7              Evaluate fitness;
8              if Chromosome is infeasible then
9                  Apply Infeasible-solution Repairing Procedure;
10                 Evaluate Fitness;
11             Parents selected using Roulette Wheel Selection;
12         for Parent chromosomes do
13             Apply uniform crossover as per P_c;
14             Mutate resulting offspring as per P_m;
15             Offspring chromosomes generated;
16         for Offspring chromosomes do
17             Evaluate fitness of new candidates;
18             Replace low-fitness chromosomes with better offspring;
19         for Each chromosome do
20             Select chromosomes for next generation;
21 Output the best fit chromosome as solution;
```

Figure 3-4: Repairing Genetic Algorithm Pseudocode

A pseudocode of the algorithm is shown in *Figure 3-4*. The chromosomes are defined for the VMs that allocate each application, as it is shown in *Figure 3-5*. Once the initial population is created using LCFP, the parents chromosomes are selected from the pool using the *Roulete Wheel Selection* algorithm detailed in [31], which basically considers the chromosomes with higher fitness to have more chances to be the parents. All the parents must be feasible. When the parents are selected, we can generate different solutions by crossing over and mutating the results. In the crossover method, a binary mask of chromosome length is created and, for each 1 contained, the chromosome is swapped with each other. Whereas, mutation method selects two exact positions and swaps the gens within the chromosome. The steps

that this algorithm follows based in a typical GA would be:

1. **Encoding**: The chromosomes have a length of the number of applications, where each gene is represented by a positive integer representing the VM assigned to this application. The gene value has a value ranging from 1 to the number of VMs.

2. **Selection**: *Roulette Wheel Selection* algorithm selects the parents chromosomes.

3. **Genetic Operations**: New population is generated by applying crossover and mutation algorithms. The mutation probability is set as a low number in order to control the search space. An example of mutation would be to change the gens of a chromosome in a 2% of possibilities.

4. **Termination**: The termination condition specifies the maximum number of generations should be reached when adequately solves the problem. Every iteration of the algorithm creates a population consisting of a set of chromosomes, which represents a possible assignment solution.

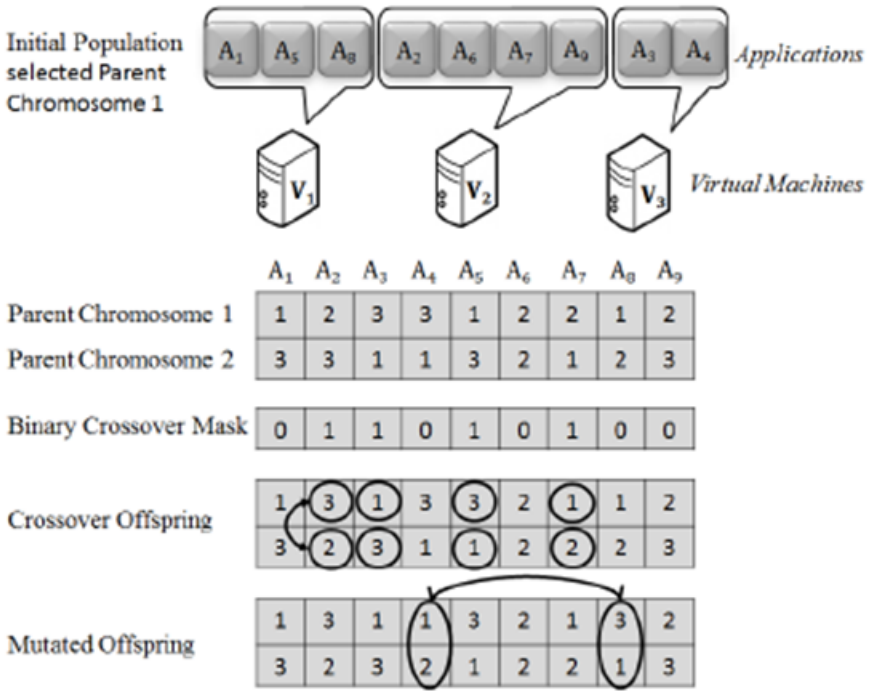

Figure 3-5: RGA Crossover and Mutation chromosomes.

The fitness function determines the quality of the current solution. A lower energy cost and higher resource utilization result in a higher solution fitness. Feasible solutions have a positive fitness value, whereas infeasible solutions incur a negative fitness. The fitness function is derived as follows:

$$F(x) = \beta_1 \cdot F^{-}{}_{obj} - \beta_2 \cdot \frac{1}{m} \cdot \sum_{j=1}^{m} \left(F_j^{cpu} + F_j^{mem} \right) \tag{3.1}$$

The weights β_1, β_2 associated with the fitness function is currently set to [2, 1]. The multiplicative inverse of the objective function discussed in the next equation is represented by $F^{-}{}_{obj}$. In order to normalize and scale the objective function F_{obj} to a range of [1, 10], it uses the next equation.

$$F^{-}{}_{obj} = \frac{F_{worst} - F_{obj}}{F_{worst} - F*} \cdot \frac{F*}{F_{obj}} \cdot range + 1 \tag{3.2}$$

where the range = 9. The best value of F_{obj} (the value that is higher minimized) and worst objective functions are represented by $F*$ and F_{worst} respectively.

The functions to ensure higher CPU and memory utilizations by penalizing constraint violations is given by:

$$F_j^{cpu} = \begin{cases} 0, & if\ U_c^{avg} = 1 \\ \text{MIPS(j)}/IC_j, & if\ 0 < U_{avg} < 1 \\ 2, & if\ U_c^{avg} = 0 \end{cases} \tag{3.3}$$

$$F_j^{mem} = \begin{cases} 2(1 - 1/\gamma), & if\ \gamma >= 1 \\ 2, & otherwise \end{cases} \tag{3.4}$$

$$\text{where } \gamma = \frac{M_j^{max}}{\sum_{j=1}^{n} \left(x_{ij} \dot{M}_r(i) \right)}$$

where the maximum capacity of a VM is declared as M_j^{max}, the requested memory of an application is $M_r(i)$, and the x_{ij} is a binary matrix which represents if an application $_i$ is allocated in the VM $_j$.

The main components of this genetic algorithm are explained in detail below.

LCFP - Generated Initial Population

The Longest Cloudlet Fastest Processor (LCFP) heuristic assigns the longest application to the fastest processing VM. Three main steps follow this heuristic:

1. Sort applications $A_i, \in I$ in descending order of execution time.
2. Sort VMs $V_j, \in J$ in descending order of processing power (MIPS).
3. Pack sorted applications into fastest processing VM.

This ensures that the lengthier applications finish quickly that should minimize the total time of execution.

IRP – Infeasible-solution Repairing Procedure

The Infeasible-solution Repairing Procedure (IRP) tries to convert infeasible solutions to feasible ones. An infeasible solution is characterized by a negative fitness from the penalty due to CPU and memory constraint violations. The idea of this component is to solve this infeasibility by reassigning some applications to different VMs and making the fitness positive. The steps that IRP follows are:

1. Find the VMs which are infeasible by CPU capacity or memory violations.

2. For each VM violated, it tries to move the applications one by one to other VMs only if these VMs have enough resources to allocate these applications. If the movements of the applications solve the violations, this VM is no longer violated. All the VMs must not be violated for making the chromosome feasible.

An example from [1] is shown in the next Figure 3-6. We can see that VM with id=1 is violated considering CPU MIPS and memory bytes used for the overall applications, which are higher than the total capacity of MIPS and memory of this VM.

The algorithm tries to move the first application A2 to others VMs starting by VM with id=3. As we can see, A2 could be moved to the VM id=3 since no violation is caused by any resource constraint. Once it has moved, the violation of VM id=1 has been solved. There are not more VMs violated so it means the infeasible solution has converted to a feasible solution one.

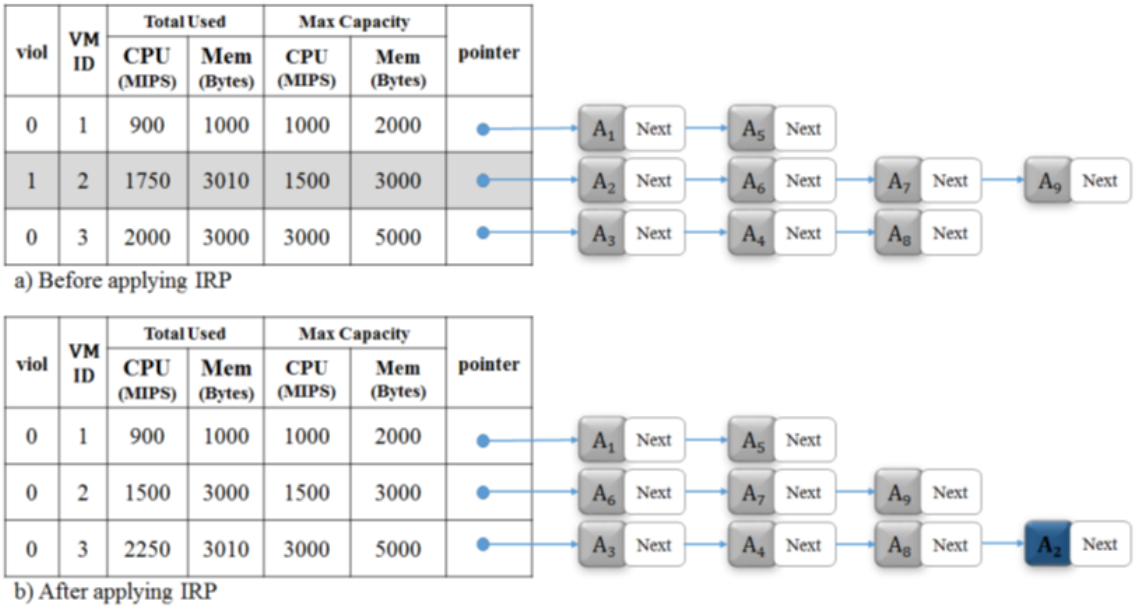

Figure 3-6: IRP solving example.

3.4.1.2 VM Placement

In this section is explained the static process when the VMs are allocated in a host for the first time. The available algorithms that this block uses are the same as considered in dynamic section *VM placement*. To avoid redundant information in the document, consider these algorithms shown in the other section as candidates to be used in this step. Unlike the dynamic step, this step is static and it is executed once and at the beginning of the simulation.

3.4.2 Dynamic

Several heuristics have been proposed in the literature for dynamic consolidation of VMs based on an analysis historical of data of the resource usage of VMs. These heuristics propose an efficient adaptive heuristics for dynamic adaption of VM allocation at run-time, according to the current utilization of resources applying live migration, switching idle nodes to the sleep mode and thus, minimizing energy consumption.

CloudSim checks in a predefined scheduling interval time if any VM could be mi-

grated between hosts. Beloglazov et al. [22] presents some adaptive heuristics to deal with the consolidations. According to [22] and following the steps that CloudSim follows, this problem can be divided in four parts consecutively in its execution timeline: (1) determine when a host is considered as overloaded requiring migration of one or more VMs from this host (*Section: Host Overload Detection*); (2) select the VMs that should be migrated from an overloaded host (*Section: VM Selection*); (3) finding a new server for these selected VMs (*Section: VM placement*); (4) determine after previous steps if any host is considered as being underloaded leading to a decision to migrate all VMs from this host, and switch the host to the sleep mode (*Section: Host Underload Detection*).

VM placement Optimization

```
1  Input: hostList  Output: migrationMap
2  foreach host in hostList do
3  |  if isHostOverloaded(host) then
4  |  |  vmsToMigrate.add(getVmsToMigrateFromOverloadedHost(host)
5  migrationMap.add(getNewVmPlacement(vmsToMigrate))
6  vmsToMigrate.clear()
7  foreach host in hostList do
8  |  if isHostUnderloaded(host) then
9  |  |  vmsToMigrate.add(host.getVmList()
10 |  |  migrationMap.add(getNewVmPlacement(vmsToMigrate))
11 return migrationMap
```

Figure 3-7: VM Placement Optimization Algorithm template

The general algorithm of VM placement optimization is shown in Figure 3-7 from [22]. First, the algorithm considers the list of hosts to apply the overloading detection algorithm, and checks whether a host is overloaded. If the host is overloaded, the algorithm applies the VM selection policy to select VMs that need to be migrated from the host. Once the list of VMs to be migrated from the overloaded hosts is built, the VM placement algorithm is invoked to find a new placement for VMs that will be migrated. The second phase of the algorithm is finding underloaded hosts, and tries to move all the VMs that the server allocates to other hosts. The algorithm returns the combined migration map that contains the information of the new VM

placements of the selected VMs to be migrated from both overloaded and underloaded hosts. The complexity of the algorithm is 2N, where N is the number of hosts.

3.4.2.1 Host Overload Detection

The first step is detecting the hosts that are overloaded. One solution is setting an upper and lower utilization thresholds which says that the total utilization of the CPU of all VMs in a host should stay between these thresholds.

However, dynamic and unpredictable workloads of different applications types make difficult to have a fixed utilization thresholds. It is desirable to find workload patterns generated by the applications. These patterns are obtained through heuristics which makes an auto-adjustment of the utilization thresholds based on a statistical analysis of historical data collected during the lifetime of the VMs. The main idea of the algorithms proposed is to adjust the value of the upper utilization threshold depending on the strength of the deviation of the CPU utilization. The higher the deviation is, the lower the value of the upper utilization threshold should be, since it is more likely that the CPU utilization will reach 100% for higher deviation values.

According to previous assumptions, these heuristics have been considered in this section: (1) Median Absolute Deviation (MAD); Interquartile Range (IRQ); Local Regression (LR); Local Robust Regression (LRR). Safety Parameter is a tuning parameter used in previous heuristics considered as the threshold that permits to tradeoff between SLA violation and resource usage.

Median Absolute Deviation

The Median Absolute Deviation (MAD) is a measure of statistical dispersion. The MAD is a robust statistic being more resilient to outliers in a data set than the standard deviation. In the standard deviation, the distances from the mean are squared, so large deviations are weighted more heavily and outliers can heavily influence it. However, in the MAD the magnitude of the distances of a small number of outliers is irrelevant. This estimator works better with distributions without a

mean or variance, such as the Cauchy distribution.

For the list of utilization CPU in a host defined as an univariate data set $X_1, X_2, ...X_n$, the MAD is defined following next formula:

$$MAD = median_i(|X_i - median_j(X_j)|) \tag{3.5}$$

The absolute deviations have been calculated from the data set median, and then MAD is obtained by getting the median of these absolute values. The upper utilization threshold is defined as:

$$T_u = 1 - s \cdot MAD \tag{3.6}$$

Where $s \in R^+$ is the safety parameter exposed at the end of the section. The maximum utilization T_u range from 0 to 1. A host is overloaded when the CPU utilization is higher than this threshold obtained.

Interquartile Range

The Interquartile range (IQR) is a measure of statistical dispersion that uses the difference of the third and first quartiles $IQR = Q_3 - Q_1$, where the univariate dataset $X_1, X_2, ...X_n$, is formed by the history of the utilization CPU in the host. This is a robust statistic which has a breakdown point (i.e. the proportion estimation of incorrect observations) of 25%. The upper threshold is defined as:

$$T_u = 1 - s \cdot IQR \tag{3.7}$$

Where $s \in R^+$ is the safety parameter exposed at the end of the section. A host is overloaded when the CPU utilization is higher than this threshold obtained.

Local Regression

The Local Regression (LR) heuristic is based on the Loess method proposed by Cleveland [32]. The main idea of the local regression method is fitting simple models to localized subsets of data to build up a curve that approximates the original data.

This heuristic is fully explained in [22] and [33], and it has been recovered briefly in this section. It considers that, for each new observation, it is possible to find a trend line used to estimate the next observation of CPU utilization known as $\hat{g}(x_{k+1})$. This is useful to detect the hosts that are overloaded, and it happens when the following inequalities are satisfied.

$$s \cdot \hat{g}(x_{k+1}) >= 1 \qquad x_{k+1} - x_k <= t_m \tag{3.8}$$

Where $s \in R^+$ is the safety parameter exposed at the end of the section, and t_m is the maximum time required of any of the VMs allocated to the host.

Robust Local Regression

The Robust Local Regression (LRR) is other heuristic proposed by the Cleveland [32], and it is fully explained in [22] and [33]. It considers that Local Regression is vulnerable to outliers that can be caused by leptokurtic or heavy-tailed distributions. For this reason, this method proposes the addition of the robust estimation that transforms the Loess into an iterative method. It follows almost the same steps as Local Regression, where the inequalities are the same explained in the previous method.

Safety Parameter

The safety parameter is a constant tuning parameter used by the previous heuristics in the percentage at scale from 0 to 1. This parameter is used to increase or decrease the upper thresholds in MAD and IQR, and it is also used in LR and LRR to deal with their respective inequalities. When the safety parameter increases, the algorithm will be safer leading to less SLA violations but higher resource wastage, and vice versa. For example, if we consider the upper thresholds of MAD, the higher safety parameter value, the lower values will be the upper thresholds giving the allocation of VMs more restrictive for the hosts. Thus, this parameter has to be tuned in order to trade-off between SLA violation and resource usage.

3.4.2.2 VM Selection

When overloaded hosts have been selected, next step is to obtain the particular VMs to migrate from these hosts. The described policies are applied iteratively, which means that, when a VM is selected to be migrated, the host is checked again if it is being overloaded. If it is still considered as overloaded, the VM selection policy is applied again to select other VMs to migrate from this host. This is repeated until the host is not overloaded.

In this section there are presented different heuristics used in this process: (1) Minimum Migration Time (MMT), which gets the VM that requires the minimum time to migrate; (2) Random Choice Policy (RC), which selects a random VM; (3) Maximum Correlation Policy (MC), which gets the VM which is higher correlated with the resource usage; (4) Minimum Utilization (MU), which gets the VMs that its CPU utilization is lower.

Minimum Migration Time

The Minimum Migration Time (MMT) gets the VM that requires the minimum time to complete the migration. The migration time is estimated as the amount of RAM utilized by the VM divided by the network bandwidth available for the host, where CloudSim defines the half of the total bandwidth of the host for migrations, and they do not use any bandwidth privilege for any VM type.

Random Choice Policy

The Random Choice Policy (RC) gets any VM of the overloaded host independently of the resource and the relation of this VM with the CPU utilization.

Maximum Correlation Policy

The Maximum Correlation (MC) policy considers that, the higher is the correlation between the resources used by the applications running on an oversubscribed server, the higher the probability of the server overloaded. According to this idea, it

selects those VMs that have higher correlation of CPU utilization than other VMs.

Minimum Utilization

The Minimum Utilization (MU) gets the VM that uses less CPU utilization. For each VM, it calculates the dedicated MIPS used for the containing applications of this VM. Once it has been done, it gets the minimum value of the overall VMs.

3.4.2.3 VM Placement

This section explains the algorithms used for finding a host for the VMs. The selected host for the VM needs to have enough resources for allocating this VM, it must not be any of the overloaded hosts obtained in *host overload detection*, and the hosts could not be overloaded once the VM is allocated.

It can be seen as a bin packing problem [34] with variable bin sizes and prices; bins represent the physical hosts; items are the VMs that have to be allocated; bin sizes are the available capacities of the hosts; and prices correspond to the power consumption by the hosts. Bin packing is a NP-hard problem, so it is not possible to find an optimal result in a reasonable time, but different approaches have been considered to deal with that problem.

Several algorithms have been proposed to deal with that problem. Some popular solutions discussed in [35] and [36] are:

1. *First Fit* (FF): Assign each VM/item to the first host/bin available.

2. *First Fit Decreasing* (FFD): The VMs are sorted in decreasing order, and then they are processed as FF.

3. *Best Fit Decreasing* (BFD): Like FFD, BFD also sorts items in decreasing order and, then, for packing these VMs, it chooses a host with minimum empty space to be left after the item is packed (e.g. this space could be the memory or the CPU utilization used in the host). Sometimes the quality of solution found by BFD could be worse than the solution found by FFD, or vice versa.

4. *Worst Fit Decreasing* (WFD): Like BFD, but it chooses a host with maximum empty space to be left after the allocation of the VM.

5. *Second Worst Fit Decreasing* (SWFD): Same as WFD, but it takes the bin with second maximum empty space.

Based in the previous popular algorithms, several authors have been working in their own algorithms and approaches. Some of them have been collected and considered, which are mainly focused in decreasing the energy consumption.

The list of algorithms are: (1) Power Aware Best Fit Decreasing, default solver used in the CloudSim package which uses a BFD strategy; (2) Guazzone et al. presents an improved BFD strategy; (3) Modified Worst Fit Decreasing VM Placement is an improvement of the WFD; (4) Second Modified Worst Fit Decreasing VM Placement is an improvement of SWFD; (5) Shi et al. approaches which includes a group of algorithms based in the BFD strategy.

Power Aware Best Fit Decreasing

Power Aware Best Fit Decreasing (PABFD) is the default algorithm that CloudSim uses in the power simulations implemented by Beloglazov et al. [22] or [33]. The VMs to migrate are sorted decreasingly by their current CPU utilizations. Then, each VM is allocated to a host that provides the least increase of power consumption caused by the allocation. This allows the VMs with higher demand to be allocated in the best hosts that increases energy consumption minimally. A pseudocode is shown in Figure 3-8, and it was obtained from the articles cited.

```
Algorithm 2: Power Aware Best Fit Decreasing (PABFD)
1  Input: hostList, vmList   Output: allocation of VMs
2  vmList.sortDecreasingUtilization()
3  foreach vm in vmList do
4      minPower ← MAX
5      allocatedHost ← NULL
6      foreach host in hostList do
7          if host has enough resources for vm then
8              power ← estimatePower(host, vm)
9              if power < minPower then
10                 allocatedHost ← host
11                 minPower ← power
12     if allocatedHost ≠ NULL then
13         allocation.add(vm, allocatedHost)
14 return allocation
```

Figure 3-8: Power Aware Best Fit Decreasing

Guazzone et al. Algorithm

This algorithm is explained in [37], and it implements a Best-Fit Decreasing strategy. The pseudocode is shown in Figure 3-8, and it was obtained from the previous reference. The steps that follows are:

1. Hosts are sorted by three sorting criteria: first, by their power status criteria, considering the first ones those that are already powered on, and then those that still powered off; second, it considers the available CPU capacity in descending order; finally, they are sorted in increasing order of idle power consumption.

2. VMs are sorted by CPU demand.

3. Iterate the sorted VMs with the sorted hosts, always considering the basic constraints of overloaded hosts, resource space and utilization thresholds.

BFD($M, V, \mathbf{C}, \bar{C}, \mathbf{s}^{max}, \hat{\mathbf{s}}, \mathbf{u}^{max}, \hat{\upsilon}(k), \omega_0$)

Input: M the set of all PMs,
V the set of powered on VMs,
$\mathbf{C}$ the vector of CPU capacities,
$\bar{C}$ the CPU capacity of the reference machine,
$\mathbf{s}^{max}$ the vector of maximum aggregated CPU share demands,
$\hat{\mathbf{s}}(k)$ the vector of predicted CPU share demands for VMs,
$\mathbf{u}^{max}$ the vector of maximum CPU utilizations,
$\hat{\upsilon}(k)$ the vector of predicted contributions to CPU utilization,
ω_0 the vector of idle power consumptions.

Output: $\mathbf{x}$ the vector of selected PMs,
$\mathbf{Y}$ the matrix of VM allocations (each element y_{ij} represents the allocation of VM j on PM i),
$\mathbf{S}$ the matrix of CPU shares for all VMs (each element s_{ij} represents the CPU share assigned to VM j on PM i).

1: $\mathbf{x} \leftarrow \mathbf{0}$
2: $\mathbf{Y} \leftarrow \mathbf{Z}$
3: $\mathbf{S} \leftarrow \mathbf{Z}$
4: $\mathbf{r} \leftarrow \min\{\mathbf{1}, \mathbf{C}\mathbf{s}^{max}\}$
5: $\mathbf{u} \leftarrow \min\{\mathbf{1}, \mathbf{u}^{max}\}$
6:
7: $M' \leftarrow \text{SORT}(M, \{\text{POWERSTATUS} : (\text{ON}, \text{OFF}), \mathbf{r} : \downarrow, \omega_0 : \uparrow\})$
8: $V' \leftarrow \text{SORT}(V, \{\hat{\mathbf{s}} : \downarrow\})$
9:
10: **for all** $v \in V'$ **do**
11: **for all** $m \in M'$ **do**
12: $\hat{s} \leftarrow \hat{s}_v(k)\frac{\bar{C}}{C_m}$
13: $\hat{\upsilon} \leftarrow \hat{\upsilon}_v(k)\frac{C}{C_m}$
14: **if** $\hat{s} \leq r_m$ **and** $\hat{\upsilon} \leq u_m$ **then**
15: $x_m \leftarrow 1$
16: $y_{mv} \leftarrow 1$
17: $s_{mv} \leftarrow \hat{s}$
18: $r_m \leftarrow r_m - \hat{s}$
19: $u_m \leftarrow u_m - \hat{\upsilon}$
20: **break**
21: **end if**
22: **end for**
23: **end for**
24: **return** $\langle \mathbf{x}, \mathbf{Y}, \mathbf{S} \rangle$

Figure 3-9: Guazzone et al. Algorithm

Modified Worst Fit Decreasing VM Placement

Chowdhury et al. [36] implements a classic Worst First Decreasing (WFD) known as Modified Worst Fit Decreasing VM Placement (MWFDVP) for power consumption purposes. The pseudocode is shown in Figure 3-10, and it was obtained from its article. The authors considered that worst-fit decisions lead to better situations in the future, instead of the more intuitive best fit decreasing algorithm.

As it is shown in the pseudocode, it considers the destination host as the candidate host that increases mostly the power consumption among all the available hosts which also have enough resources to allocate the VM.

Modified Worst Fit Decreasing VM Placement (MWFDVP)

```
1.  Input: hostList, VMList  Output: allocation of VMs
2.  VMList.sortByCpuUtilization_decreasing()
3.  foreach VM in VMList do
4.          maxPower = Double.MIN_VALUE
5.          allocatedHost = null
6.          foreach host in hostList do
7.              if host has enough resources for VM
8.                  powerAfterAllocation =
    getPowerAfterAllocation(host,VM)
9.                  powerDiff = powerAfterAllocation  –
    host.getPower()
10.                 If powerDiff > maxPower
11.                     maxPower = powerDiff
12.                     allocatedHost = host
13.         If allocatedHost ≠ null then
14.             allocation.add(VM,allocatedHost)
15. Return allocation
```

Figure 3-10: Modified Worst Fit Decreasing VM Placement

Second Modified Worst Fit Decreasing

Chowdhury et al. [36] implements a classic Second Worst First Decreasing (SWFD) known as Second Modified Worst Fit Decreasing VM Placement (SWFDVP) for power consumption purposes. The pseudocode is shown in Figure 3-11. The authors consider that worst-fit decisions lead to better situations in the future instead of the more intuitive best fit decreasing algorithm. As it is shown in Figure 3-11, it considers the destination host as the second host that grows mostly the power consumption among all the available hosts which also have enough resources to allocate the VM.

```
Second Worst Fit Decreasing VM Placement (SWFDVP)
1.  Input: hostList, VMList  Output: allocation of VMs
2.  VMList.sortByCpuUtilization_decreasing()
3.  foreach VM in VMList do
4.          maxPower = Double.MIN_VALUE
5.          allocatedHost = null
6.          secondHost=null
7.          foreach host in hostList do
8.              if host has enough resources for VM then
9.                  powerAfterAllocation =
    getPowerAfterAllocation(host,VM)
10.                 powerDiff = powerAfterAllocation –
    host.getPower()
11.                 If powerDiff > maxPower
12.                     maxPower = powerDiff
13.                     If(allocatedHost!=null)
    secondHost=allocatedHost
14.                     allocatedHost = host
15.         If secondHost ≠ null then
16.             allocation.add(VM,secondHost)
17.  Return allocation
```

Figure 3-11: Second Worst Fit Decreasing VM Placement

Shi et al.

Shi et al. [38] implements some algorithms based in the Best Fit Decreasing (BFD) approach. They mainly use two metrics: the magnitude for the VMs, which are calculated using the CPUs (CPU_v) and memory ($Memory_v$) required for each VM, shown in Equation (3.9); the capacity for the hosts, which are calculated using the CPUs (CPU_i) and memory ($Memory_i$) that they contain, shown in Equation (3.10). They explore six different algorithms fully explained in [38], where we recovered two of them: *Percentage Utilization*, which means that PMs are sorted in decreasing order based on the ratio of the absolute capacity of the PM against the total magnitude of the VMs hosted on it; *Absolute Capacity*, the PMs are decreased in the order of the absolute capacity measure.

$$Rq(v) = \sqrt{(CPU_v)^2 + (Memory_v)^2} \tag{3.9}$$

$$Cp(i) = \sqrt{(CPU_i)^2 + (Memory_i)^2} \tag{3.10}$$

The pseudocode is shown in Figure 3-12, and it was obtained from its article. It decreases the candidate hosts by a concrete magnitude, and then the VMs are allocated to each of these hosts that have enough resources. The VMs are sorted in decreasing order by their magnitudes.

Algorithm

Input: v, P, M
Output: M

```
1: for i ← 1 to length[P] do
2:     Calculate the occupied magnitude O[i]
3: end for
4: Sort P in descending order of occupied magnitude
5: for i ← 1 to length[P] do
6:     if v can fit into P[i] then
7:         place v onto P[i]
8:         update O[k] of PM k, M
9:         Sort P in descending order of occupied magnitude
10:        return;
11:    end if
12: end for
13: reject request of v
```

Figure 3-12: Shi et al. Algorithm

3.4.2.4 Host Underload Detection

This section explains the last step of the dynamic VM consolidation process. The approach of CloudSim is the only one considered.

CloudSim Host Underload Detection

This is the default algorithm that CloudSim power package uses in the migration process. It finds the host with minimum utilization and tries to place all the VMs from this host to other hosts keeping them not being overloaded and accomplishing resource constraints. If this can be accomplished, the VMs are set for migration, and the host is switched to the sleep mode once all migration have been completed. This process is iteratively repeated for all hosts that have not been considered as being overloaded.

Chapter 4

Architecture

In this chapter, the overall architecture and components designed with the purpose of this project are presented. In the first section, the *conceptual model* is discussed, giving the main idea of the project where the relevant elements are linked. In the second section, the *optimization process* is detailed, showing how the datacenter works to optimize its energy consumption based in the algorithms considered in the previous chapters. Finally, the *infrastructure* where we executed the experiments is introduced.

Taking into account the technologies available, we selected CloudSim v4.0. It has a wide recognition framework in the research area that it generates datacenters simulations to study in several areas like the one that interest us: energy optimization. It does not have the enough low-level detail as others frameworks discussed in the document, but it gives facilities to adhere the implementation of new algorithms, besides the large literature that it has. This simulator is easy and flexible which helps to implement the scheduling algorithms, and it perfectly fulfilled our requirements. Last version of CloudSim was selected because it contained a more stable framework and included the power package functionalities.

RStudio open-source software was selected to evaluate the data [39]. It has several facilities to integrate machine learning libraries and analyze the data, besides the familiarity that we have with them. The amount of data used allows us to avoid the use of distributed execution using frameworks such as Apache Hadoop or Apache Spark that follow batch oriented approaches.

4.1 Conceptual Model

For the purpose of this project, there are two main components previously introduced: CloudSim and RStudio. Initially, the idea of this project is to generate different simulations and then analyze them. The first step, it is to give real workload traces to simulate the loading of the tasks, and we generate the corresponding applications in CloudSim. The environment is configured considering the number of hosts, VMs and their corresponding properties for each simulation generated. Each simulation generates a file log explaining the details of the execution (i.e. the actors of the migrations, the state of the hosts in a determined time, etc.) and other file in CSV format with the result and the final instance of the simulation. To avoid generate some huge logs in terms of size, it is implemented the option of disable the detailed log of the simulation.

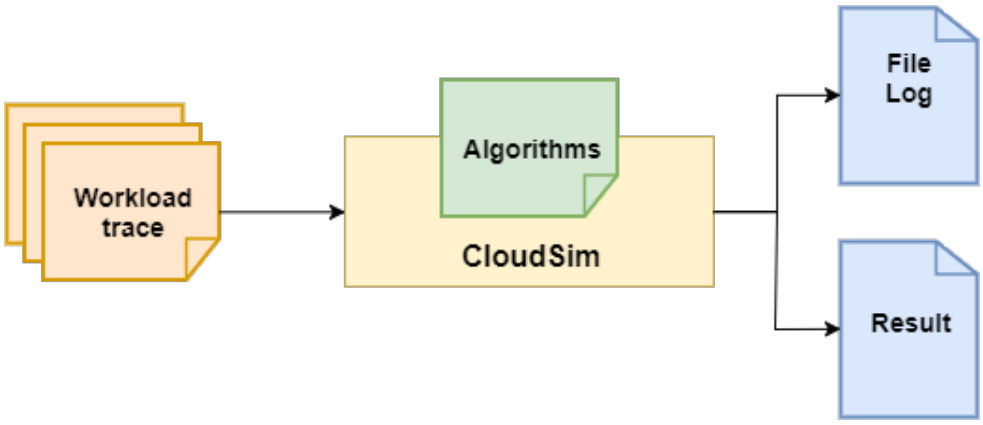

Figure 4-1: CloudSim Simulation results process

Once we have generated the resulting instances, we use RStudio to study and predict the response variable that is the energy consumption. We build different models by using the 75% of instances for training and 25% of instances for testing, and we compare their effectiveness.

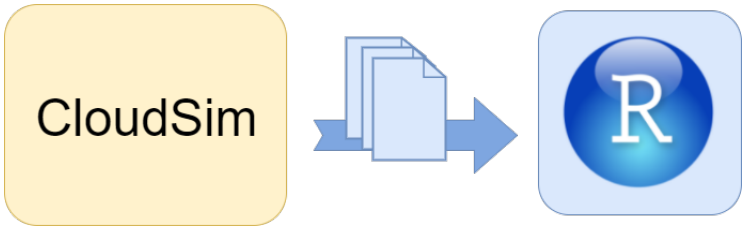

Figure 4-2: CloudSim and RStudio environments

Finally, given a scenario, the properties of the tasks and the algorithm applied as an input data, the model will predict the energy consumption as it is shown in Figure 4-2.

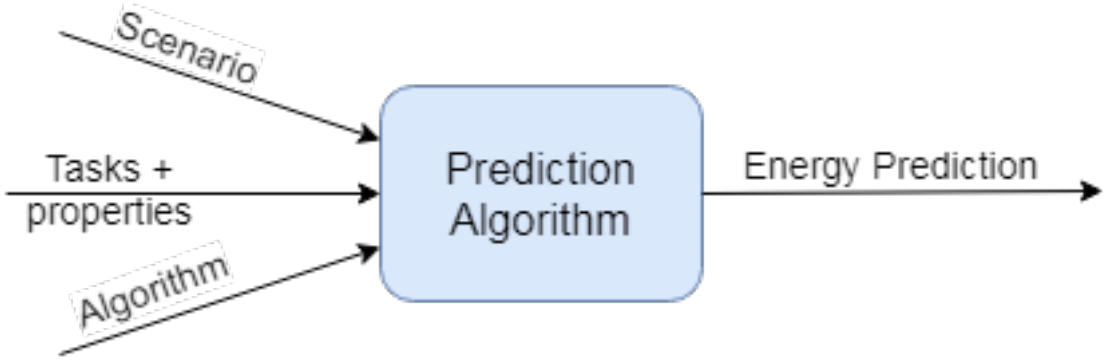

Figure 4-3: Attributes and Prediction Algorithm linked

4.2 Optimization process

The optimization process implemented in the project that was partially reutilized from the CloudSim power package is shown in Figure 4-4. Static and dynamic process are identified, where static means that this block is executed once, and dynamic means that it is executed several times. In this case, the static steps are executed at the beginning of the simulation. The dynamic steps are executed iteratively following a scheduling interval predefined. Similarly, there is another scheduling interval that defines when the processing of cloudlets assigned in a VM are updated. This last scheduling interval checks, for each interval, if the tasks have already finished. The algorithms and heuristics that can be used in each step are detailed in *Algorithms*.

The process steps are detailed in the next list:

Input Data

1. Workload of tasks have been generated using external data or patterns. The configuration of the experiment defines the number of hosts and VMs in addition to their corresponding characteristics.

Static

1. All the tasks are assigned to the VMs considering Task Placement complexity.
2. VMs are assigned to the hosts considering VM Placement complexity.

Dynamic

1. Hosts overloaded have been detected considering principally the loaded CPU.
2. For each host overloaded, some VMs have been selected to migrate to other hosts.
3. For each VM, it is necessary to find a new candidate host. The candidate host must not be overloaded and need to have enough resources to allocate the VM. This is considered as the VM Placement complexity.
4. Finally, it is checked if any host is underloaded, which means that contains few VMs to be migrated to others hosts without causing the destination host an overloaded situation.
5. Repeat the step 1 after each scheduling interval time predefined. The simulation is completed when all cloudlets instructions have been processed correctly.

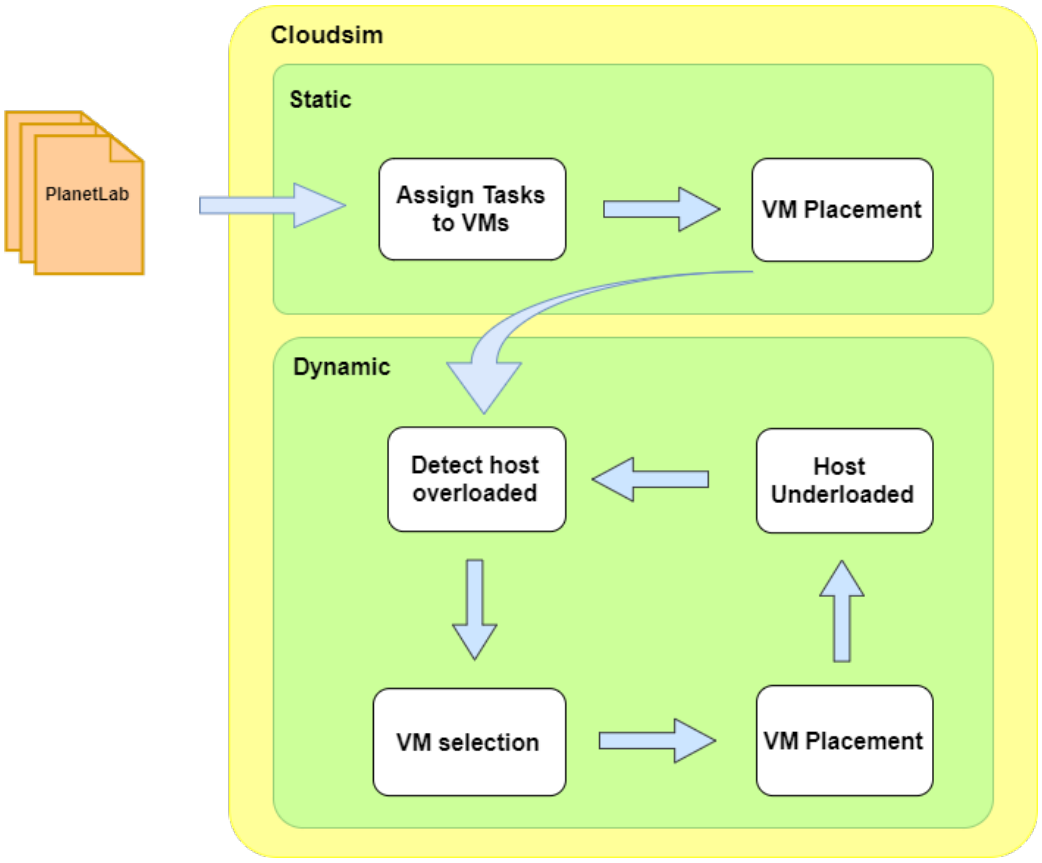

Figure 4-4: Optimization Process in CloudSim.

4.3 Infrastructure

In this section we discuss about the physical infrastructure on top of which we have conducted our experiments. The infrastructure was provided by the Research and Development Laboratory (RDLab) [40], a department of the UPC. The purpose of this group is to provide comprehensive IT support to the Computer Science department of UPC and other research centers in order to optimize project's research and development processes. The High Performance Cluster (HPC) from RDLab consists of more than 1000 CPU cores and more than 3TB of RAM memory. These are shared between multiple projects. The execution environment also offers support for Lustre (a parallel distributed file system), Hadoop Mapreduce, Apache Spark, SMP and MPI computation and GPU computing among others.

The cluster works using jobs, where each job contain several tasks to be executed. This job can be executed in one determined node if it is desired, allowing the configuration and the properties of the job requirements as could be the memory or the CPU cores needed. The jobs can also be stopped on demand. The nodes can be accessed remotely via Secure Shell (ssh), which is a cryptographic network protocol, for job execution purposes.

The generation of the simulations in CloudSim is explained in the *experimental desing* that is considered. This task only finds a suitable environment which gives the result in a reasonable time, only focusing in getting the output and not considering the total time (i.e. the performance) that needs the executions. The implementation of the CloudSim algorithms and their associated generator of experiments are being exported in one JAR . Then, this file is executed in one machine selected among all the available machines in the cluster. It has been proved that machines with two cores improve effectively the execution, while more CPUs will be almost useless for the performance. The memory is adapted with the size of the problem that it is wanted to simulate. We have generated several JARs to distribute the executions among different machines of the cluster.

Chapter 5

Experimental Study

This chapter conducts experiments to demonstrate energy-efficiency and quality of solutions based in selected algorithms using the *optimitzation process*. Some models based in the experiments are generated to learn and predict the system. Datacenters are continuously running applications, and it is common to find datacenters that have high or low loading in different days. For this reason, two scenarios are considered.

This chapter is structured as follows:

- *Attributes*: different aspects have been exported of the datacenter simulations to further analysis.
- *Experimental Design*: characteristics of the experiments in terms of tests sets, algorithms, applications and VMs attributes among others.
- *Modeling Design*: preprocessing, machine learning algorithms and evaluation criteria in the datasets.
- *Scenario #1*: analysis and modeling of a scenario which considers an empty datacenter with incoming applications.
- *Scenario #2*: analysis and modeling of a scenario which considers a loaded datacenter with incoming applications.

5.1 Attributes

This section shows the selected attributes in Table 5-1 obtained from the CloudSim simulations. We could divide the attributes considering entity types and objectives, such as the applications, VMs, hosts, configured simulation system or response variables. Some attributes are affected by various entities, and the most predominant entity has been selected. The classification would be:

- Applications: NUM_CLOUDLETS, MEAN_APP_LENGTH, SD_APP_LENGTH, MEAN_APP_MEMORY, SD_APP_MEMORY
- VMs: NUM_VMS, APPS_VMTYPE0, APPS_VMTYPE1, APPS_VMTYPE2, APPS_VMTYPE3, NUM_VM_MIGR, NUM_VM_MIGR_TYPE0, TIME_VM_MIGR, NUM_VM_MIGR_TYPE1, NUM_VM_MIGR_TYPE2, NUM_VM_MIGR_TYPE3
- Hosts: MEAN_RESOURCE_UTILIZATION, MAX_RESOURCE_UTILIZATION
- Simulation System: VM_PLACEMENT_POL, SIMULATION_TIME, SLA, SCENARIO
- Response variables: ENERGY_AVG, ENERGY_CONS, POWER_SUM

Attribute Name	Type	Comments
NUM_VMS	Integer	Number of VMs.
NUM_CLOUDLETS	Integer	Number of Applications.
MEAN_APP_LENGTH	Integer	Mean Cloudlet Length.
SD_APP_LENGTH	Integer	Standard Deviation Cloudlet.
MEAN_APP_MEMORY	Integer	Mean Cloudlet Memory.
SD_APP_MEMORY	Integer	Standard Deviation Memory.
APPS_VMTYPE0	Double	Number of tasks assigned to VMs type 0.
APPS_VMTYPE1	Double	Number of tasks assigned to VMs type 1.
APPS_VMTYPE2	Double	Number of tasks assigned to VMs type 2.
APPS_VMTYPE3	Double	Number of tasks assigned to VMs type 3.
MEAN_RESOURCE_UTILIZATION	Double	The percentage mean of the memory usage of the active hosts.
MAX_RESOURCE_UTILIZATION	Double	The maximum percentage of the memory usage of the active hosts.
VM_PLACEMENT_POL	String	VM Placement Policy.
SIMULATION_TIME	Double	Time execution of the experiment in seconds, also known as makespan.
POWER_SUM	Double	The total power consumed.
ENERGY_AVG	Double	The energy consumed in Watts per second.
ENERGY_CONS	Double	The energy consumed in kWh.
SLA	Double	Service Level Agreement
NUM_VM_MIGR	Double	Number of Migrations
TIME_VM_MIGR	Double	Time used in migrations in seconds.
NUM_VM_MIGR_TYPE0	Double	Number of migrations of VMs of type 0.
NUM_VM_MIGR_TYPE1	Double	Number of migrations of VMs of type 1.
NUM_VM_MIGR_TYPE2	Double	Number of migrations of VMs of type 2.
NUM_VM_MIGR_TYPE3	Double	Number of migrations of VMs of type 3.
SCENARIO	Integer	Scenario used.

Table 5.1: Experiment Attributes

5.2 Experimental Design

The experimental design is divided in some relevant aspects. These aspects are: CloudSim configuration and policies; parameters settings for applications, VMs and hosts; tests sets determining the size of the experiment; and the optimization algorithms and heuristics considered. Finally, it is shown the number of instances that each scenario uses.

CloudSim configuration and policies

The Cloudlet policy selected is the *Dynamic Workload* because it is the policy specially dedicated for power purposes in CloudSim. The VM policy selected is the *Time Shared Over Subscription* shown in *Policies*. Due to the fact of unpredictable

changes of the CPU of the tasks, it is safer and recommended to use the policy which protects the simulation when the CPU usage surpasses the 100%. When this happens, it redistributes the usage of the MIPS among all the VMs of the host and the execution will not fail.

Parameters settings for applications, VMs and Hosts

Applications have relevant parameters to be considered in the experiments: CPU usage, Millions of Instructions (MI) and maximum memory capacity (MaxMemTask).

The CPU percentage usage is filled by a real data center workload shown in *datasets*. PlanetLab traces are selected, which are configured that every five minutes of the execution is updated the CPU usage according to their traces.

To generate different experiments isolating MI and MaxMemTask, the mean and the standard deviation of these variables values are considered. It is assumed that they follow a Gaussian distribution pseudo-random values that contemplate different resources values making a stable and repeatable experiment. For each variable, we followed the next considerations:

Millions of Instructions

Millions of Instructions (MI) refers to the size of the application to be processed. The minimum value for this variable is 400.000, and it is increased by 400.000 until the amount reaches the maximum value of 8.000.000. For the standard deviation, we considered the minimum value of 30.000, and it is increased by 30.000 until the amount reaches the maximum value of 90.000. All these combinations (20 different mean lengths and 3 standard deviations) are considered to analyze variations in the application's length size. We considered that these combinations could happen in a real environment.

Maximum Memory Capacity

Maximum Memory Capacity (MaxMemTask) is the maximum memory required for the application (i.e. it is not the memory used in the execution time) and, con-

sequently, once this value is calculated, it will not change during the execution time, simplificating the complex problem of the memory variables. It is considered three levels according to the resources requirements: high, medium and small types shown in Table 5.2. Units in MB.

Application Memory Levels		
Level	Mean	sd
High	750	100
Medium	500	100
Small	250	100

Table 5.2: Parameter settings for applications

Once the mean and standard deviation of the application length and the memory task level are generated in one determined test set, these values will not change for the evaluated algorithms. It is assumed that the applications only use one PE or CPU.

Four types of VM have been considered in our experiments, depending on their properties. VM characteristics are based on Amazon EC2 instance types [41] with the only exception that the VMs are single-core. It is shown in Table 5-3.

VM Type	MIPS	Ram (MB)	Bandwidth	PEs	Size (MB)
Large	2500	870	100Mbit/s	1	2500
Medium	2000	1740	100Mbit/s	1	2500
Small	1000	1740	100Mbit/s	1	2500
Micro	500	613	100Mbit/s	1	2500

Table 5.3: Parameter settings for VMs

It is assumed that the CPU required for a VM must be single-core. Following the same reasoning as [22], if the CPU capacity required for a VM is higher than the capacity of a single core, then a VM must be executed on more than one core in parallel. However, it is not assumed that VMs can be arbitrarily parallelized, as there is no a priori knowledge of the applications running on a VM and automatic parallelization is a complex research problem. The bandwidth of all VMs types are 100Mbit/s, where each task uses the 10% of this VM bandwidth. The proportion of VM types is the same to ensure that the study is balanced in the allocation of applications.

Three types of Hosts have been considered in our experiments, reusing partialy the configuration defined by Beloglazov et al. [22] shown in Table 5.4. Power consumption characteristics of these servers are shown in Table 5.5, where it is showed the consumption in Watts for each CPU percentage used in the server.

Power Model	MIPS	Ram	Bandwidth	Storage	PEs
HP ProLiant ML110G3	1200	4096	1000000 GB/s	1TB	2
HP ProLiant ML110G4	1860	4096	1000000 GB/s	1TB	2
HP ProLiant ML110G5	2760	4096	1000000 GB/s	1TB	2

Table 5.4: Parameter settings for Hosts

The proportion of hosts types is the same to ensure that the study is balanced in the allocation of VMs.

Servers	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant ML110 G3	105	112	118	125	131	137	147	153	157	164	169
HP ProLiant ML110 G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant ML110 G5	93.7	97	101	105	110	116	121	125	129	133	135

Table 5.5: Servers characteristics of the experiments

Test Sets

10 test sets have been considered in our experiments to examinate various experiment sizes. The number of applications range from 15 to 1400 with their corresponding number of VMs shown in Table 5.6.

Test Set	VMs	Applications
1	12	15
2	24	30
3	52	60
4	100	125
5	152	180
6	300	350
7	500	600
8	752	900
9	1000	1200
10	1200	1400

Table 5.6: Test sets for the experiments

We assigned the number of VMs multiple of four to have the same proportion of VM types. The number of hosts for each test set is assigned to 2100. This environment simulates many servers that they do not have enough processing capacity compared with the incoming applications size, but it allows using enough machines to process the incoming applications. The longer a task is running in the data center, the more intensive the scenario will be, giving the chance for the datacenter to explore the selected algorithms.

The number of hosts is multiple of three to have the same proportion of type hosts. Only one datacenter is considered to study the static and dynamic assignations, without considering network complexities that would be another focus of the problem.

Algorithms and Heuristics

To consider the interactions between the algorithms considered in the *optimization process* without doing too many combinations, some decisions of that are shown below.

- For the static assignments:
  - Task Placement: Repairing Genetic Algorithm.
  - VM Placement: PABFD, MWFDVP, Shi-Pu, Shi-Ac and Guazzone.
- For the dynamic assignments:
  - Detection Host Overloaded: LR with a safety parameter of 1.2.
  - VM Selection: MMT.
  - VM Placement: PABFD, MWFDVP, Shi-Pu, Shi-Ac and Guazzone.
  - Detection Host Underloaded: CloudSim default approach.

Thus, the algorithms analyzed are primarily those which are in the VM Placement. Static and dynamic assignment of the VM placement block use the same algorithm for each execution. Repairing Genetic Algorithm has been proved to be a relevant algorithm as discussed in [1], which this implementation acts as a non-preemptive scheduling, which means that the tasks are not interrupted until they have finished, and these tasks don't change from the VM that initially execute them. The selection of the rest of the algorithms are based in [22], where this study decided that the best VM selection algorithm was MMT, and the best host overloaded algorithm was LR with the safety parameter of 1.2. in the reduction of energy consumption. It does not mean that, in other scenarios, could be better others *algorithms* described, but this is a criteria of selection to avoid the crossing of algorithms of different processes. To select some relevant VM placement algorithms, it has been considered the study of [26], where they analyzed in different perspectives some algorithms mainly focused in reducing the energy consumption. We have selected those that have better results and relevancy in this study.

Instances generated

For each scenario, the algorithms have been crossed with the test sets, applications and VM parameters settings already shown, generating the data for further analysis. The number of instances generated of each scenario would be:

Number of Test Sets	10
Number of Applications Length	20
Number of Standard Deviation Application Length	3
Number of Memory Levels	3
Number of Algorithms	5
Combinations of previous considerations	9000 instances.

Table 5.7: Instances characteristics for each scenario.

Each simulation is executed four times to guarantee the validation of the instance. We considered a relevant number of instances for modeling, keeping in mind that some of these executions could involve almost an hour to be processed.

Evaluation Criteria

The evaluation criteria of the different scenarios discussed in the project is the same for each of them. When discussing about the quality and efficiency of the solutions, some main indicators and criteria are considered:

- **Scalability**: Large datasets could require linearly or non-linear processing time in terms of the size of the problem.
- **Energy Consumption**: The total power consumption of the simulation.
- **Time**: The total time used to generate the simulation in terms of the number of applications processed.
- **Resource memory efficiency**: The resource memory efficiency used during the execution in the active servers.
- **SLA**: Violations occurred in the simulation.

5.3 Modeling Design

This section shows the modeling design used in the scenarios. The evaluation criteria and what steps used to predict the data are described in detail.

Initial Assumptions

Two approaches and datasets are considered in each scenario: a complete dataset that starts using the variables generated in the simulation; a simple dataset that only uses the applications variables as the predictors to model, trying to predict the energy consumption using the input data of the system without considering the processed variables of the simulation. Both use a traditional preprocessing, cleaning and modeling steps.

Each dataset follows the next steps: (1) a preprocessing, which selects the most relevant variables, and prepare the data for the study; (2) the exploration, which analyzes some insights of the data considering that some of these explorations are limited because they are already described in *experimental design*; (3) modeling, where some machine learning algorithms are used to predict the response variable. Finally, some conclusions of both datasets are shown.

Preprocessing

The preprocessing is composed by: (1) check the near-zero variance of the candidates predictors; (2) use the Boruta package [42], a wrapper feature selection, to select the predictors that are not interesting; (3) apply statistical significance test [43] to decide what variables are removed.

The statistical significance test is responsible to decide if any attribute is important for the modeling. It checks if the predictor is irrelevant for the energy prediction. It considers the predictor significant if the p-value returned is lower than the threshold i.e. 5%. A p-value with NA means that it could be calculated from others variables,

which it means that it is redundant or insignificant for the prediction. The iterative test process checks one by one these predictors, eliminates the worst predictor and then, it repeats the process until all the predictors are significant. The reason of following this method is that some non-significant variables could be significant when one variable is removed. When the variables are removed, the model is simpler and easier to predict even if information is lost.

A global preprocessing for all the datasets is applied. The predictors that can produce noise and redundancy are removed. Taking into acount the attributes, we discard ENERGY_AVG, POWER_SUM and SCENARIO, because the first and second one could be calculated from ENERGY_CONS and SIMULATION_TIME, and the third is a constant variable for all the instances being the scenario executed giving no extra information. We know that empty values are not presented in the datasets, so this step is not considered in the preprocessing.

Machine Learning Algorithms

Once the data is generated, the data is shuffled randomly to ensure that test and train partitions contain data of different test sets. Also the train and test instances are selected randomly with a 75% and 25% of them respectively. The response variable of the model is the energy consumption (ENERGY_CONS), which is a regression prediction derived for their continuous values.

The prediction algorithms used in the datasets are:

- **Linear Model**: It uses a linear regression which finds the best-fitting line through the observed data.

- **Log-Linear Model**: This method uses a log function for the response variable keeping the others independent variables in their original scale. These models are typically used when the variables may have an exponential growth relationship.

- **Random Forest**: Non-linear estimator that fits a number of decision tree classifier on various sub-samples of the dataset, and use averaging to improve the predictive accuracy and control over-fitting. This is also known as an ensemble method.

Evaluation Criteria

The metrics used for the evaluation of these algorithms are shown below.

Root Mean Squared Error

Root Mean Squared Error (RMSE) is a quadratic scoring rule. It is the square root of the average of squared differences between prediction and actual observation. RMSE is a negatively-oriented-scores, which means lower values are better.

If we denote y as the observation, and $\hat{y}$ the prediction, the formula would be:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \tag{5.1}$$

Mean Absolute Error

Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions without considering their direction. It is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. It is a negatively-oriented-scores, which means lower values are better.

If we denote y as the observation and $\hat{y}$ the prediction, the formula would be:

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \tag{5.2}$$

Coefficient of Determination

Coefficient of Determination, goodness of fit and known as R-squared (R^2), is a statistical measure of how close the data are to the fitted regression line. It is the percentage of the response variable variation that is explained by only linear models. It ranges from 0 to 1. The higher this coefficient, the model is better.

If we denote y_i as the observed values of the dependent variable, and y_i as its mean, and as the fitted value, then the coefficient of determination is:

$$R^2 = \frac{\sum(\hat{y}_i - \overline{y})^2}{\sum(y_i - \overline{y})^2} \tag{5.3}$$

Comparison and interpretation

All previous indicators are well-known in the literature for the regression. RMSE should be more useful when large errors are particularly undesirable because they are more penalized than MAE. For example, if we have two prediction errors of 10 and 5 values, RMSE will consider the error of 10 more than twice worse than the error of 5. Otherwise, MAE will consider the error of 10 just twice as bad of 5.

However, MAE is better for an interpretation standpoint. If we have few big errors in the prediction, RMSE will give us a bad result, while MAE will consider the absolute deviation of the prediction and the real value. Both are useful and complementary keeping in mind they advantages and drawbacks. Goodness of fit shows another perspective of how good is the linear model used.

5.4 Scenario #1: Empty datacenter

In this scenario is considered an empty datacenter, which means that no application is running at the beginning of the simulation. The incoming applications are the only considered for the analysis. The size of the experiment is calculated by considering the multiplication of number of VMs (m) with the number of applications (n).

5.4.1 Scalability

The scalability of the scenario is logarithmic as it is shown in Figure 5-1. Each point represents a test set that is calculated by applying the mean of the simulation time of overall simulations of this test set. A high increment in the first five test sets is produced, but then it is normalized. The reason is that the complexity of the applications used in the infrastructure generate important overheads, specially in the first test sets.

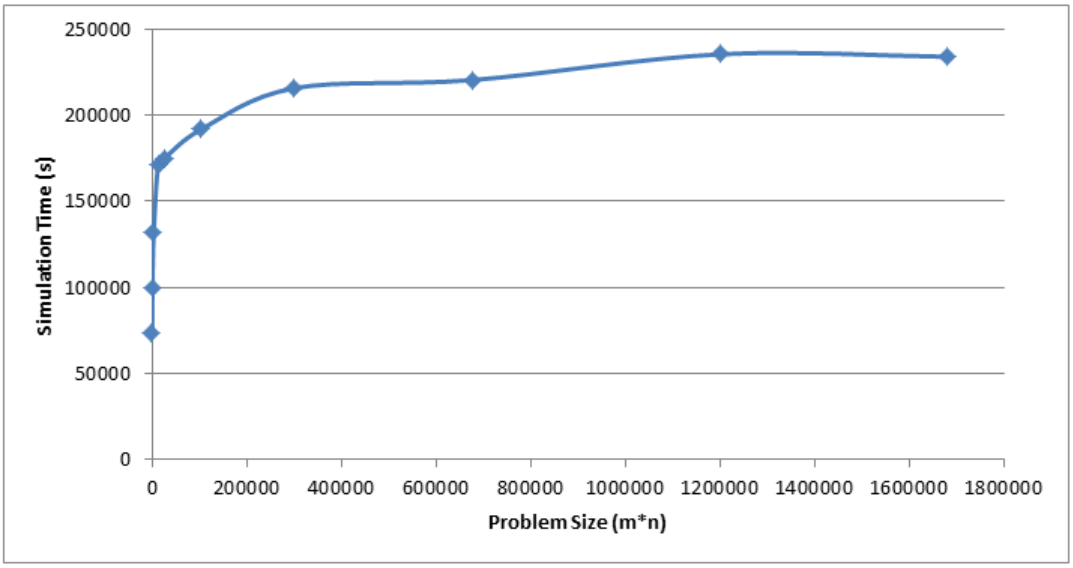

Figure 5-1: Scalability of scenario #1

5.4.2 Energy Consumption

In this section the energy efficiency in the algorithms is shown. We calculate the mean for the overall energy in each test set and algorithm.

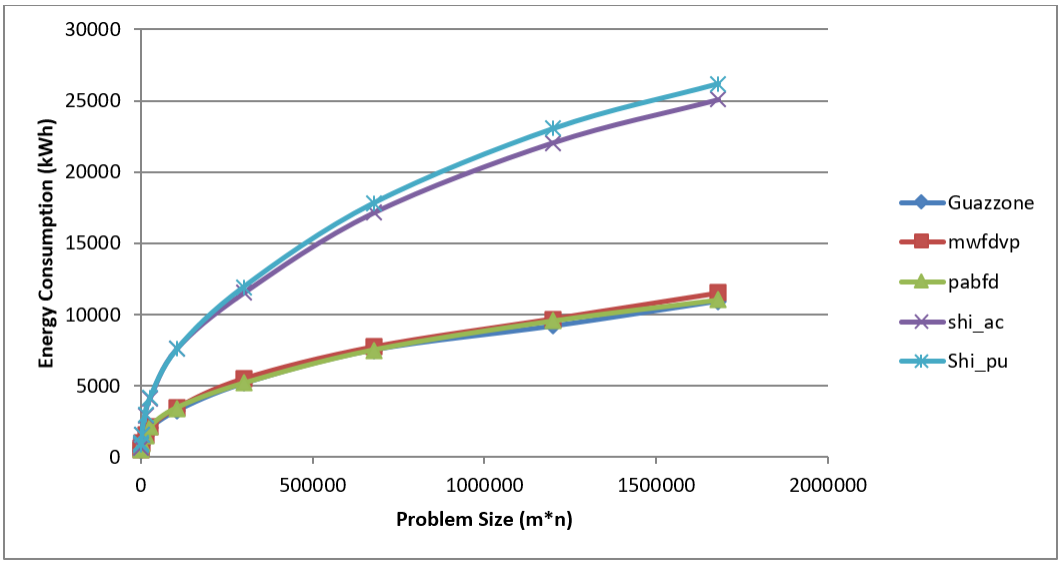

Figure 5-2: Energy Consumption in Scenario #1

Figure 5-2 shows that Guazzone is the one that gives better results followed very closely by the algorithm PABFD and MWFDVP. However, ShiAC and ShiPU obtain a high energy consumption when the test size has increased.

5.4.3 Time

In this section is discussed the time used in the simulations for each algorithm.

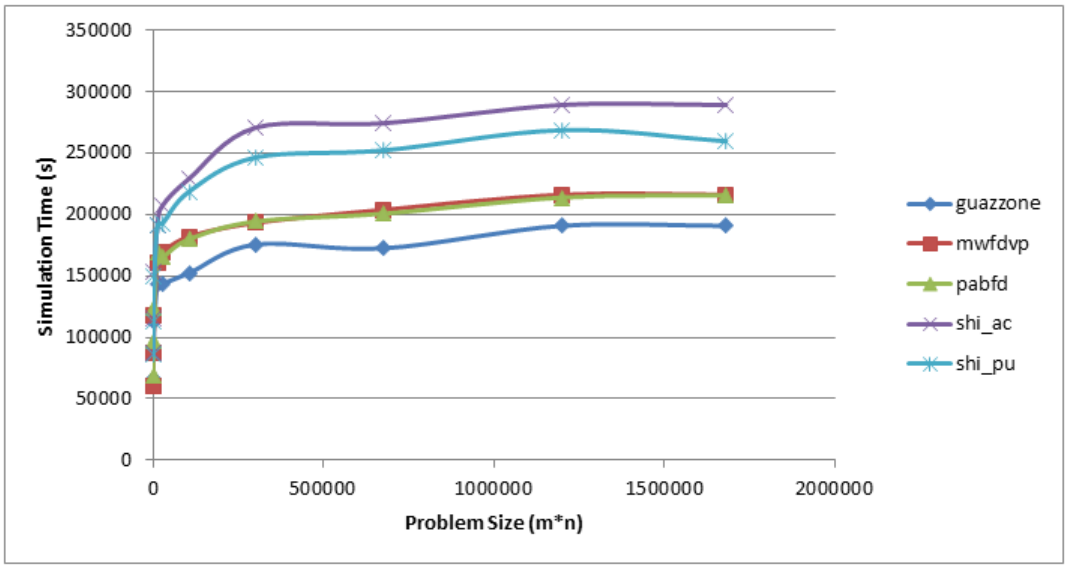

Figure 5-3: Total time in Scenario #1

Figure 5-3 shows that Guazzone algorithm improves significantly the performance when compared to other candidates. Shi approaches do not obtain good results, while PABFD and MWFDVP obtain intermediate and similar results.

5.4.4 Quality of Service

In this section the algorithms are evaluated considering the SLA, which is the metric to check if the contract between client and provider is guaranteed.

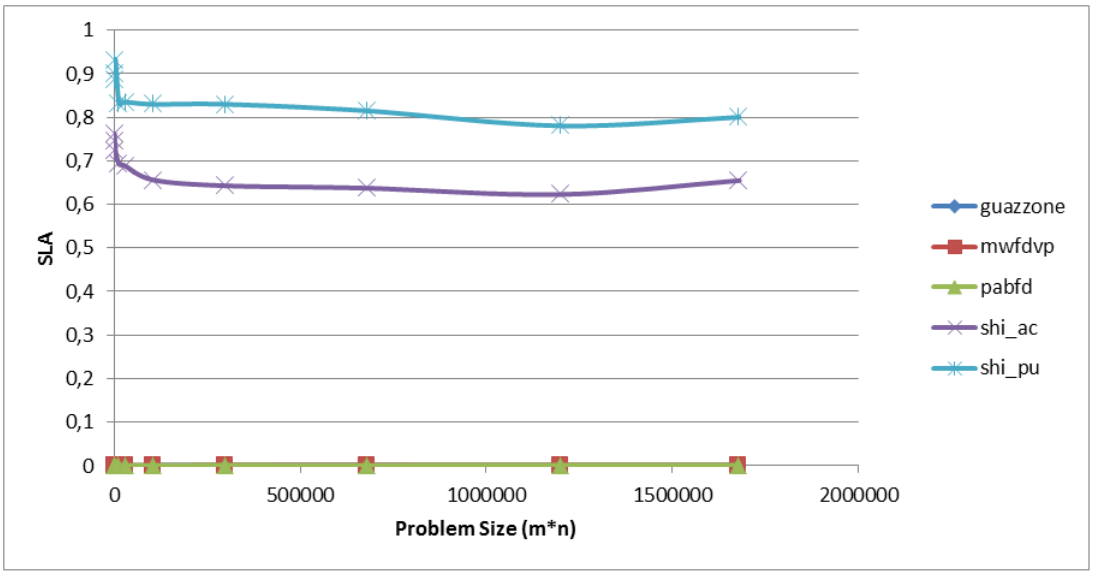

Figure 5-4: SLA in Scenario #1

Figure 5-4 shows that PABFD, Guazzone and MWFDVP guarantee an excellent SLA, while Shi approaches obtain bad results.

5.4.5 Migrations

This section discusses the main statistics of migrations. Figure 5-5 and Figure 5-6 display the number of migrations and their corresponding times. It increases strongly when the size of problem changes. Althogh the time used for migrating is clearly lower than the total execution time, these number of migrations could produce overheads in performance. We should know if the overall selected algorithms are affected by this high number of migrations.

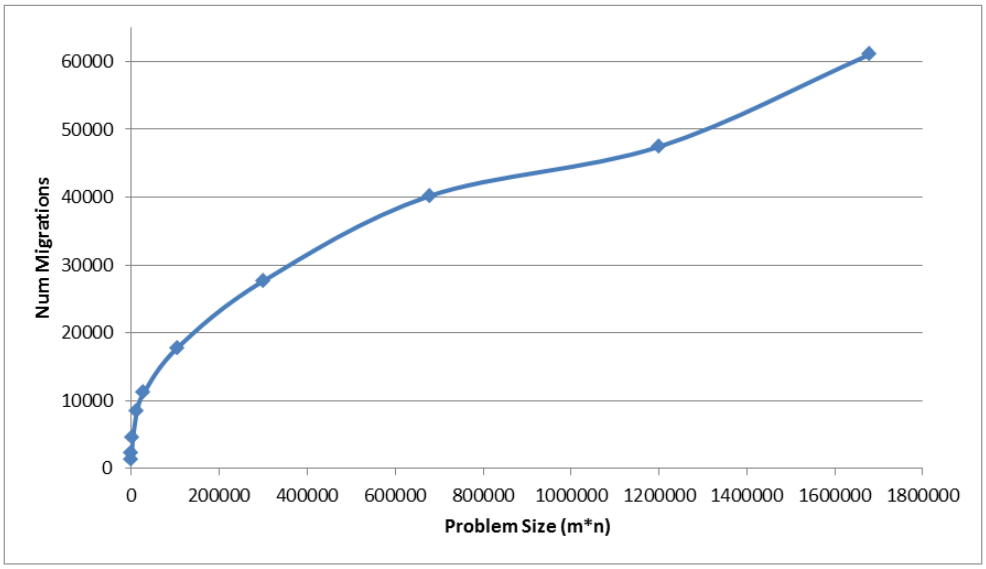

Figure 5-5: Number of migrations in Scenario #1

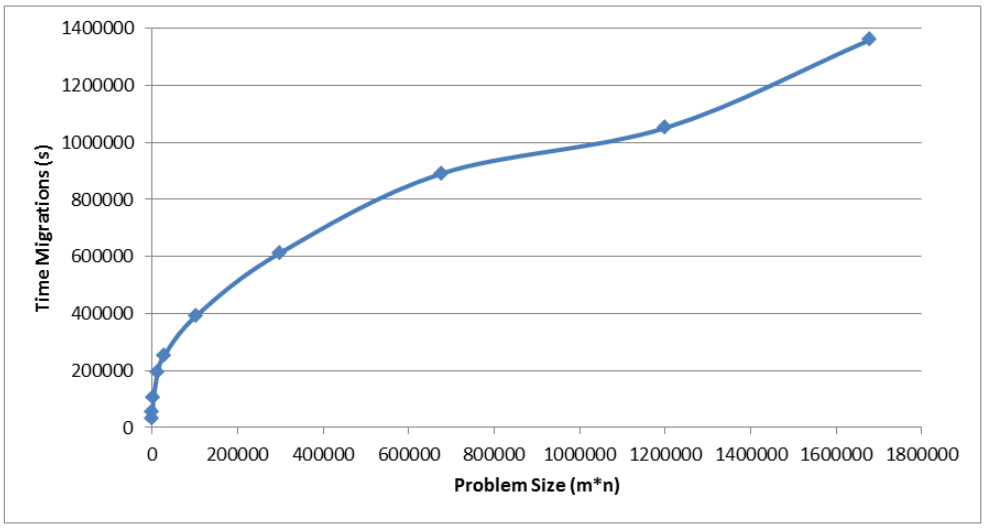

Figure 5-6: Time of migrations in Scenario #1

Figure 5-7 shows a comparison of the number of migrations between algorithms. Shi approaches obtain a high number of migrations, showing why Shi approaches obtain strange values in the experiment, while the rest have made few migrations. These number of migrations affect other properties such as the SLA, the performance or the energy consumption. PABFD, MWFVP and Guazzone consider that moving VMs towards other hosts will not improve the performance due to their estimates, even though the servers could be overloaded.

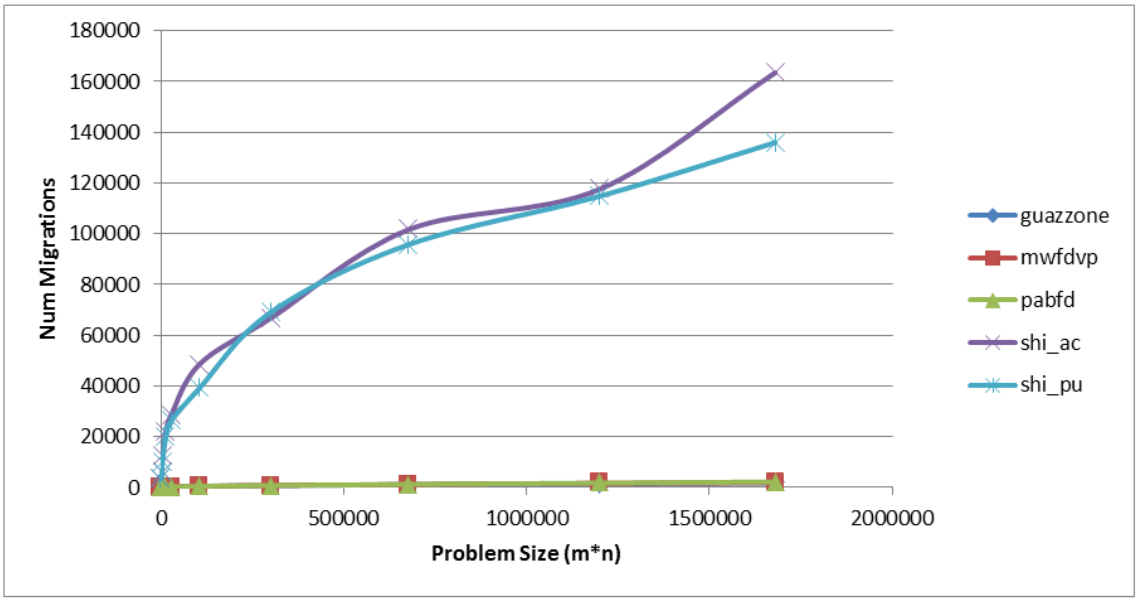

Figure 5-7: Number of migrations per algorithm in Scenario #1

Figure 5-8 shows the most used VM types in the migrations. The percentage of the VM types range from 0 to 1, which represents the most used types in each test set. All the VMs types are used during the simulations, and there is a significant difference between the migrations of VM types 1 and 2 and the rest. These VMs are those with higher RAM and, consequently, they contain more tasks running inside that produce higher CPU variations according to the trace file of each task, giving more chances to be migrated.

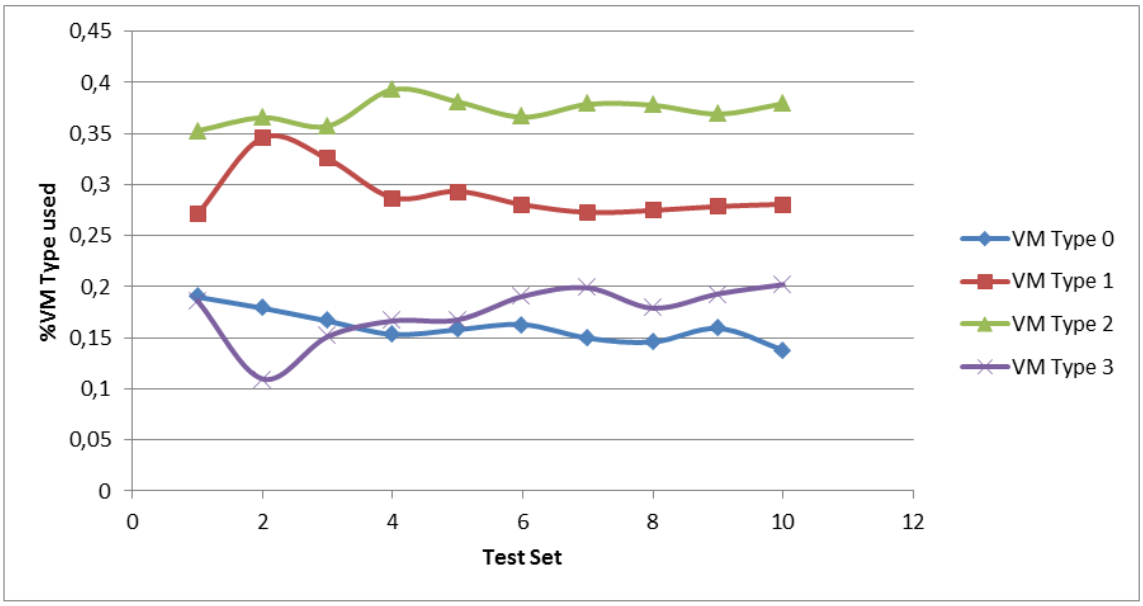

Figure 5-8: VM Types migrated in Scenario #1

5.4.6 Resource memory efficiency

In this section is shown the resource memory efficiency of the active hosts in the datacenter. All the candidates start in the initial test sets around 50% of the memory utilization. However, Guazzone, MWFDVP and PABFD grow closer when the test size problem is increased up to 70% of the utilization, while Shi approaches decrease together until 30%.

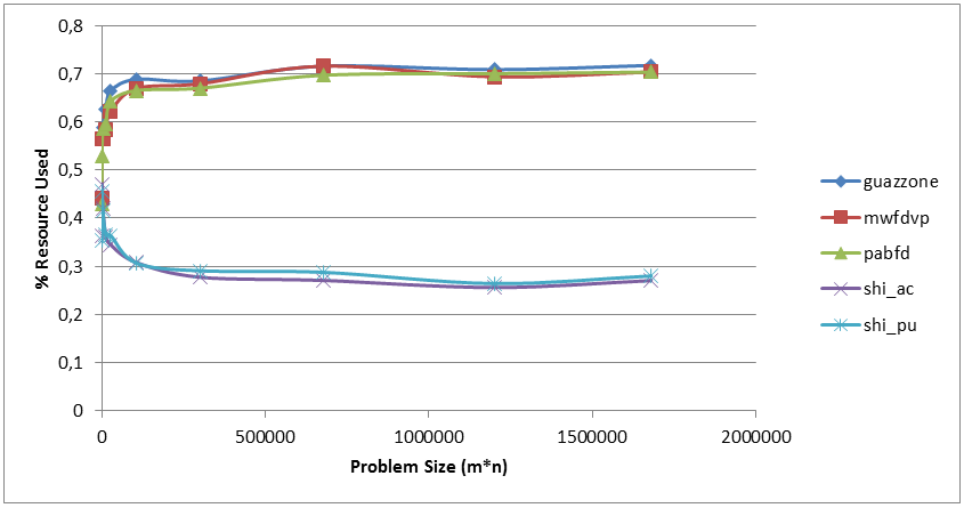

Figure 5-9: Resource memory efficiency in Scenario #1

Figure 5-10 shows the maximum resource usage in a specific moment for the overall active hosts of the datacenter. All candidates obtain good results. However, when the test set is increased, PABFD, MWFDVP and Guazzone grow their resource usage until a maximum capacity near from its maximum, while Shi approaches decrease slightly their maximum capacity.

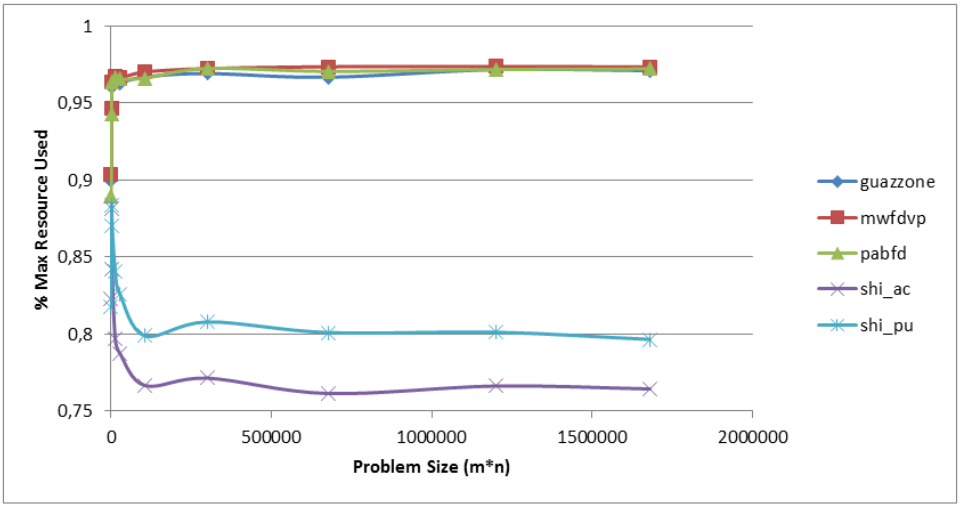

Figure 5-10: Maximum Resource Usage in Scenario #1

5.4.7 Applications Distribution on VM types

Figure 5-11 shows the VMs types that process more applications. The percentage of VM types range from 0 to 1, which represents the most used types in each test set. The most used is the VM type 1, which is the VM with the highest RAM and interesting CPU capacity. The second position is for VM type 0, which has the highest CPU capacity but little RAM, and the third position is for VM type 2, which has an intermediate CPU capacity and RAM. The last position is VM type 3 that has little RAM and CPU capacity. It is shown that application assignment is applied as expected because CPU and memory capacities are considered for the assignation.

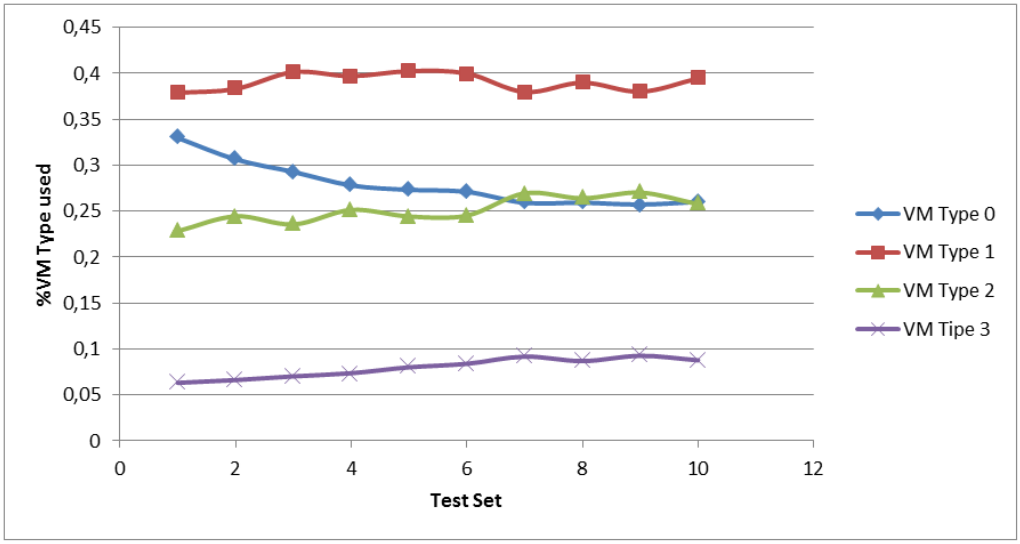

Figure 5-11: Applications Distribution in VM types in Scenario #1

5.4.8 Modeling and Prediction

5.4.8.1 Complete Dataset

Preprocessing

We checked the variables with a zero variance test that results with SD_APP_MEMORY. We used Boruta and it finds that SD_APP_MEMORY and SD_APP_LENGTH are not relevant predictors as well. Finally, the statistically significant test decided the next variables to be removed: NUM_CLOUDLETS, SD_APP_MEMORY, APPS_VMTYPE3, NUM_VM_MIGR_TYPE2, NUM_VM_MIGR_TYPE3, SIMULATION_TIME and SD_APP_LENGTH.

Exploration

The exploration of data is necessary to understand the type of data generated. The summary of the preprocessed data is shown in Figure 5-12, where some conclusions are introduced.

Particularly, we can see that:

- NUM_VMS, MEAN_APP_LENGTH, MEAN_APP_MEMORY, VM_PLACEMENT_POL are created following the known *experiment design*.
- MEAN_RESOURCE_UTILIZATION shows scattered values due the minimum, median, mean and maximum they have. The minimum and 1st quartile shows that some executions have a very bad resource utilization, while the 3rd quartile and maximum value show a good resource utilization. Both quartiles do not differ too much of the mean.
- MAX_RESOURCE_UTILIZATION shows that most instances have a good resource utilization because the majority of values are closer to 1.
- ENERGY_CONS, SLA, APPS_VMTYPE0, APPS_VMTYPE1, APPS_VMTYPE2, NUM_VM_MIGR, TIME_VM_MIGR, NUM_VM_MIGR_TYPE0 and NUM_VM_MIGR_TYPE1 are clearly unbalanced. The median is much lower than the 3rd quartile, and specially the maximum value. The main reason would be the variations of the experiment size or the algorithms applied.

```
    NUM_VMS       MEAN_APP_LENGTH   MEAN_APP_MEMORY  APPS_VMTYPE0     APPS_VMTYPE1     APPS_VMTYPE2 MEAN_RESOURCE_UTILIZATION
 Min.   :  12.0   Min.   : 400000   Min.   :250     Min.   :   0.0   Min.   :   0.0   Min.   :  0   Min.   :0.0350
 1st Qu.:  52.0   1st Qu.:2300000   1st Qu.:250     1st Qu.:  16.0   1st Qu.:  23.0   1st Qu.:  9   1st Qu.:0.3260
 Median : 226.0   Median :4200000   Median :500     Median :  66.0   Median :  94.0   Median : 49   Median :0.5410
 Mean   : 409.2   Mean   :4200000   Mean   :500     Mean   : 127.2   Mean   : 189.0   Mean   :127   Mean   :0.5113
 3rd Qu.: 752.0   3rd Qu.:6100000   3rd Qu.:750     3rd Qu.: 199.0   3rd Qu.: 327.2   3rd Qu.:207   3rd Qu.:0.6860
 Max.   :1200.0   Max.   :8000000   Max.   :750     Max.   :1073.0   Max.   :1400.0   Max.   :758   Max.   :0.8770
 MAX_RESOURCE_UTILIZATION VM_PLACEMENT_POL  ENERGY_CONS           SLA            NUM_VM_MIGR       TIME_VM_MIGR
 Min.   :0.3770           guazzone:1800    Min.   :  330.8   Min.   :0.000000   Min.   :     2   Min.   :     28
 1st Qu.:0.8370           mwfdvp  :1800    1st Qu.: 1066.4   1st Qu.:0.000052   1st Qu.:   139   1st Qu.:   2953
 Median :0.9450           pabfd   :1800    Median : 3360.6   Median :0.001085   Median :  1048   Median :  22643
 Mean   :0.8995           shi_ac  :1800    Mean   : 6372.1   Mean   :0.305726   Mean   : 22207   Mean   : 494268
 3rd Qu.:0.9750           shi_pu  :1800    3rd Qu.: 9231.9   3rd Qu.:0.664323   3rd Qu.:  9426   3rd Qu.: 199141
 Max.   :1.0000                           Max.   :75564.7   Max.   :2.537635   Max.   :621379   Max.   :14818821
 NUM_VM_MIGR_TYPE0 NUM_VM_MIGR_TYPE1
 Min.   :     0    Min.   :     0
 1st Qu.:    32    1st Qu.:    40
 Median :   224    Median :   300
 Mean   :  3339    Mean   :  6231
 3rd Qu.:  2290    3rd Qu.:  3220
 Max.   :122979    Max.   :202223
```

Figure 5-12: Summary of the preprocessed dataset in Scenario #1

Due to the fact that the majority of variables are continuous except VM_PLACEMENT_POL, a correlation matrix is processed as it is shown in Figure 5-13. Some indications are included next.

- NUM_VMS, NUM_CLOUDLETS, APPS_VMTYPE0, APPS_VMTYPE1, APPS_VMTYPE2 and ENERGY_CONS are clearly correlated. The higher test set is produced, the greater number of applications are executed in the most likely candidates VMs types.

- ENERGY_CONS is quite correlated with NUM_VMS, APPS_VMTYPE0, APPS_VMTYPE1, APPS_VMTYPE2 and the migrations variables. The greater number of migrations or test set produced, the higher energy consumption obtained.

- MEAN_RESOURCE_UTILIZATION and MAX_RESOURCE_UTILIZATION are correlated. It means that the instances which have a high mean of resource utilization also obtain a high maximum value. However, they are negatively correlated with ENERGY_CONS and the migrations variables. It seems that the migrations produce a reduction of the resource utilization, while a better usage of resource utilization results in an improvement of the energy efficiency.

- SLA and MEAN_RESOURCE_UTILIZATION or MAX_RESOURCE_UTILIZATION are very negatively correlated. The higher the resource utilization, the less SLA. It seems that some algorithms used in the implementation produce high SLA and low resource utilization scenarios, and viceversa.

- NUM_VM_MIGR, TIME_VM_MIGR, NUM_VM_MIGR_TYPE0 and NUM_VM_MIGR_TYPE1 are highly correlated. The higher number of VMs migrated, the more time will be consumed and, consequently, the number of VMs types migrated are increased as well.

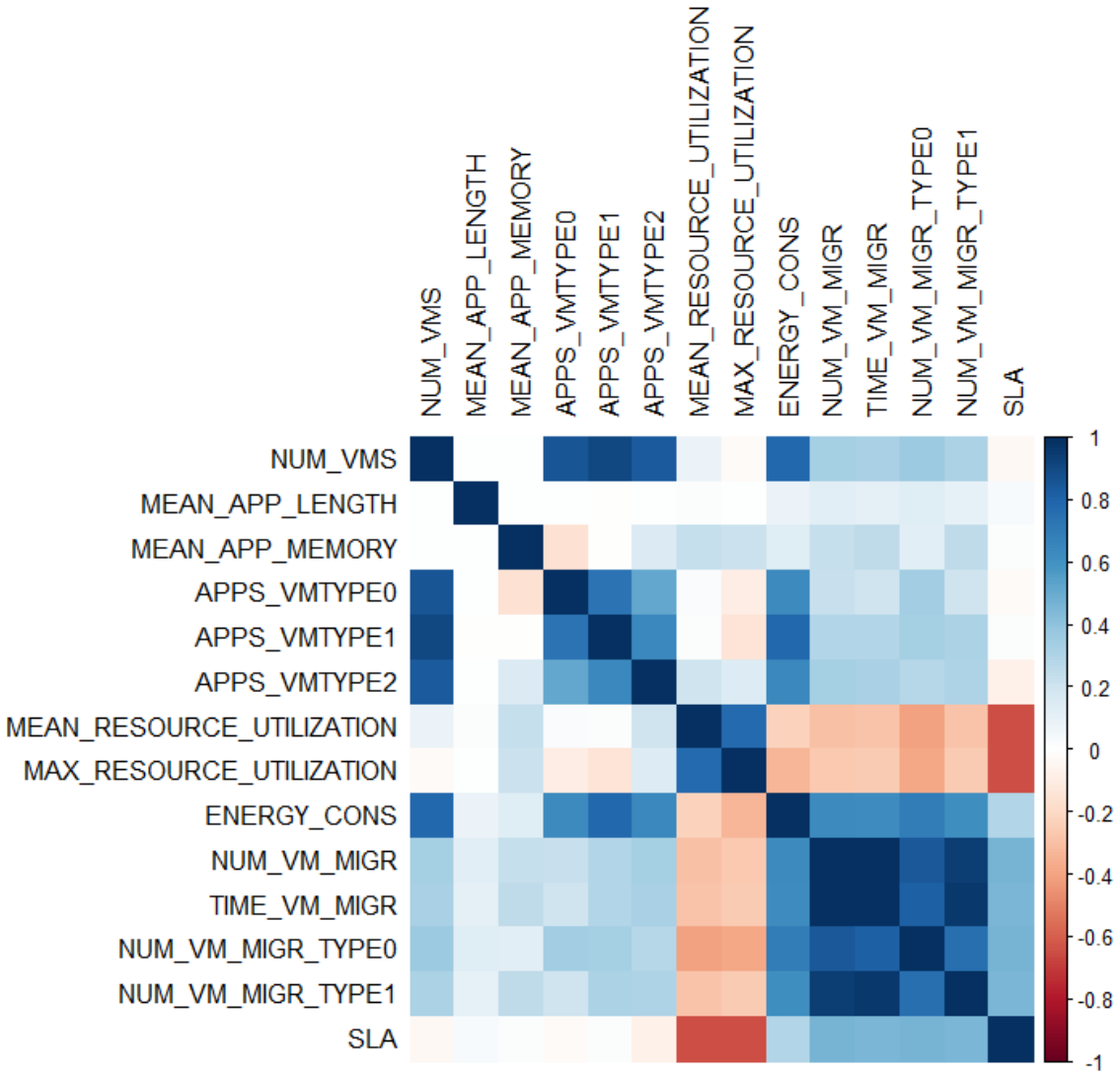

Figure 5-13: Correlation matrix of the preprocessed dataset in Scenario #1

Modeling

The comparison of the quality of the prediction methods used in the complete dataset are shown in Table 5-8. Non-linear methods produce better results than linear ones. Log-linear model produces bad results considering the mean of ENERGY_CONS, since the MAE is almost higher than 2000 and RMSE is close to 3500. Linear model produces better results than the previous one, but it moves between errors from 1500 to 2400, which it is clearly a bad model. Both algorithms obtain an error rate between 25% and 54%.

Random Forest produces a MAE almost lower than 600, and RMSE almost lower than 1000, which seems a good improvement. It obtains an error rate between 9% and 16%.

	RMSE	MAE	R2
Linear Model	2314.95	1592.17	0.89
Log-Linear Model	3434.42	1998.34	0.76
Random Forest	1057.63	602.48	-

Table 5.8: Machine Learning Algorithms comparison in the complete dataset in Scenario #1

5.4.8.2 Simple Dataset

Preprocessing

We checked variables with a zero variance that results with SD_APP_MEMORY. We used Boruta and it finds that SD_APP_MEMORY and SD_APP_LENGTH are not relevant predictors as well. Finally, the statistically significant test select the next variables to be removed: SD_APP_MEMORY and SDD_APP_LENGTH.

Exploration

The exploration is shown in the complete dataset, where the resulted variables in this dataset should be analyzed considering the summary statistics and correlation matrix already obtained in the previous dataset.

Modeling

The comparison of the quality of the prediction methods used in the simple dataset that are shown in Table 5-9. Linear methods produce very bad results considering the mean of ENERGY_CONS, producing an error rate of 34%-55%.

Random Forest produces a MAE lower than 1000, but the RMSE is higher than 1000. The error rate is between 14% and 25%, and the model does not seem very good.

	RMSE	MAE	R2
Linear Model	3536.73	2590-29	0.77
Log-Linear Model	3364.18	2183.21	0.79
Random Forest	1619.09	933.49	-

Table 5.9: Machine Learning Algorithms comparison in the simple dataset in scenario #1

5.4.8.3 Conclusions

We can conclude that the selected non-linear algorithm gives better results than other linear methods used. Complete dataset gives better results than the simple dataset, where there is a difference of more than twice in the error metrics. It seems reasonable because in the complete dataset more variables are used, which helps the model in predicting more accurately the response variable. The random forest used in the complete dataset is the most relevant model since we have errors around 9%, and considering the unbalanced values of the dataset, it seems a reasonable result. All methods consider a RMSE higher than MAE, where indicates that some predictions are quite bad, but the majority of these predictions have a small deviation with the observed value.

5.5 Scenario #2: Loaded datacenter

In this scenario a loaded datacenter is considered, which means that some applications are already running in the datacenter when incoming applications arrived. The characteristics of the hosts and the applications running in the datacenter are shown below:

- 1/3 of hosts are loaded creating equally different types of VMs. These VMs are distributed also equally to the different host types.
- The application has a constant CPU usage of 20%.
- The mean application length used for the incoming applications is the same for all the applications that are being executed in the datacenter.

- The maximum memory application used for the incoming applications is the same for all the applications that are being executed in the datacenter.
- All these applications use the 10% of the VM bandwidth.

5.5.1 Scalability

The scalability of this scenario is logarithmic as it is shown in Figure 5-14. Each point represents a test set that is calculated by applying the mean of the simulation time of overall simulations of this test set. A high increment is shown in the first five test sets, but then it is normalized with a linear increase. The reason is that the complexity of the applications used in the infrastructure generate important overheads, specially in the first test sets.

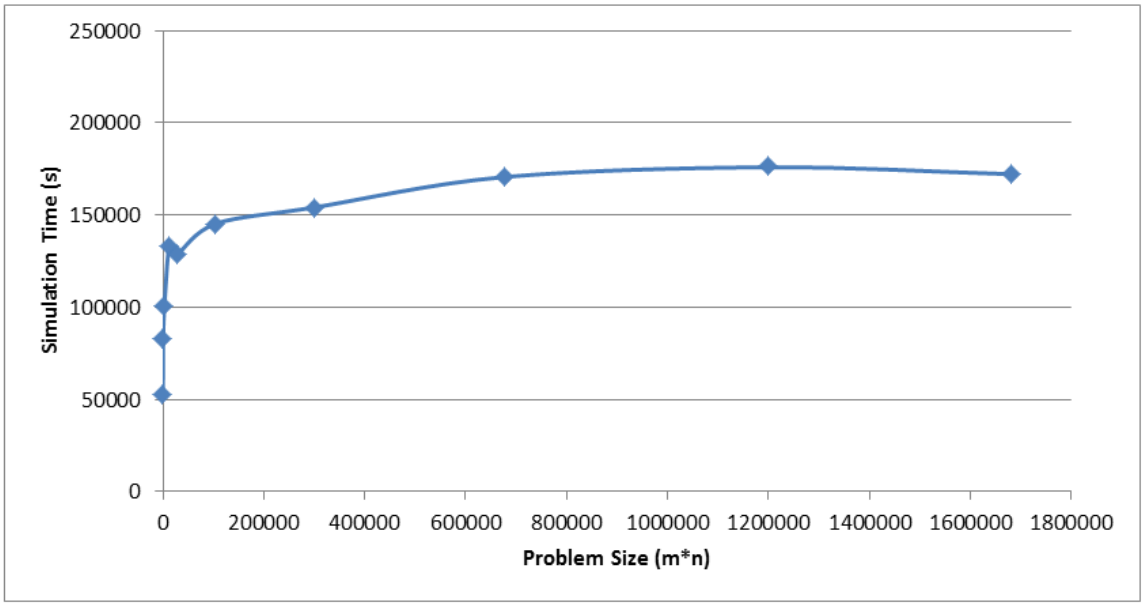

Figure 5-14: Scalability of scenario #2

5.5.2 Energy Consumption

In this section the energy efficiency of the algorithms is shown. We calculate the mean for the overall energy in each test set and algorithm.

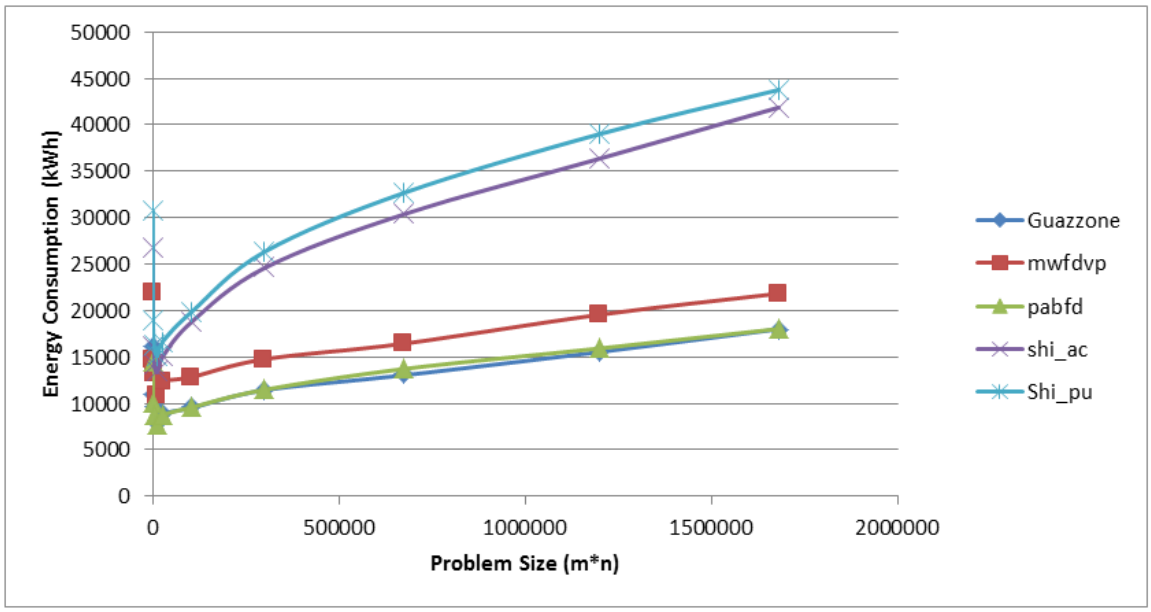

Figure 5-15: Energy Consumption in Scenario #2

Figure 5-15 shows that Guazzone and PABFD obtain the best results. MWFDVP grows linearly consuming more energy than the previous algorithms. ShiAC and ShiPU obtain high energy consumption when the test size is increased.

5.5.3 Time

Figure 5-16 shows the comparison of times between algorithms. PABFD and Guazzone obtain the best results among the other candidates. Shi approaches need almost twice as much time compared to the previous ones.

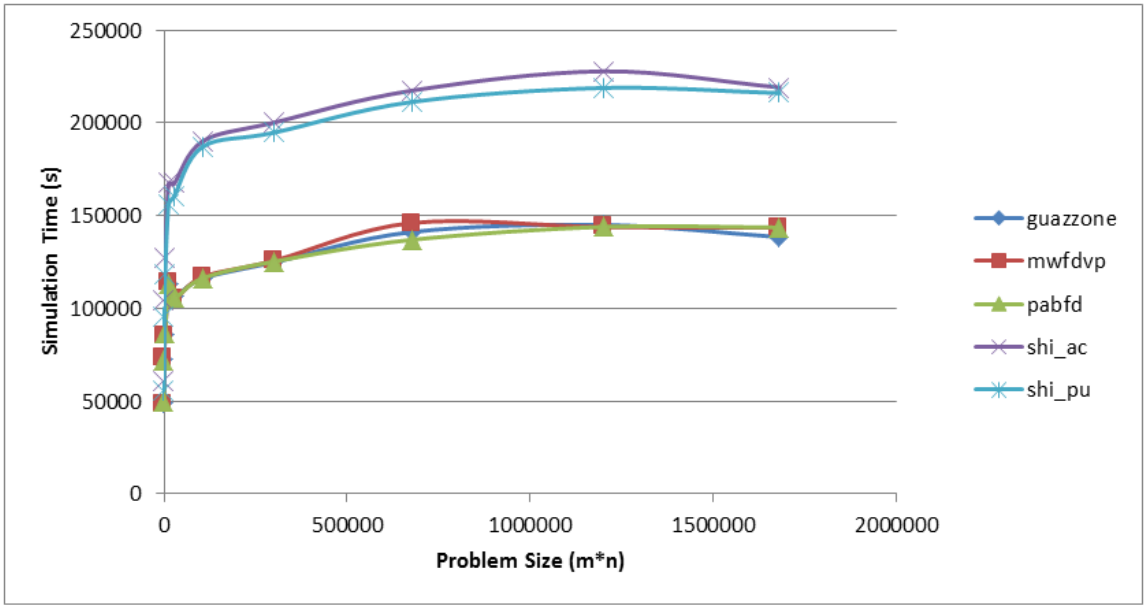

Figure 5-16: Total time in Scenario #2

5.5.4 Quality of Service

Figure 5-17 displays the comparison of QoS between algorithms. PABFD, MWFDVP and Guazzone produce nearly 0 violations, while Shi approaches produce a high number of violations.

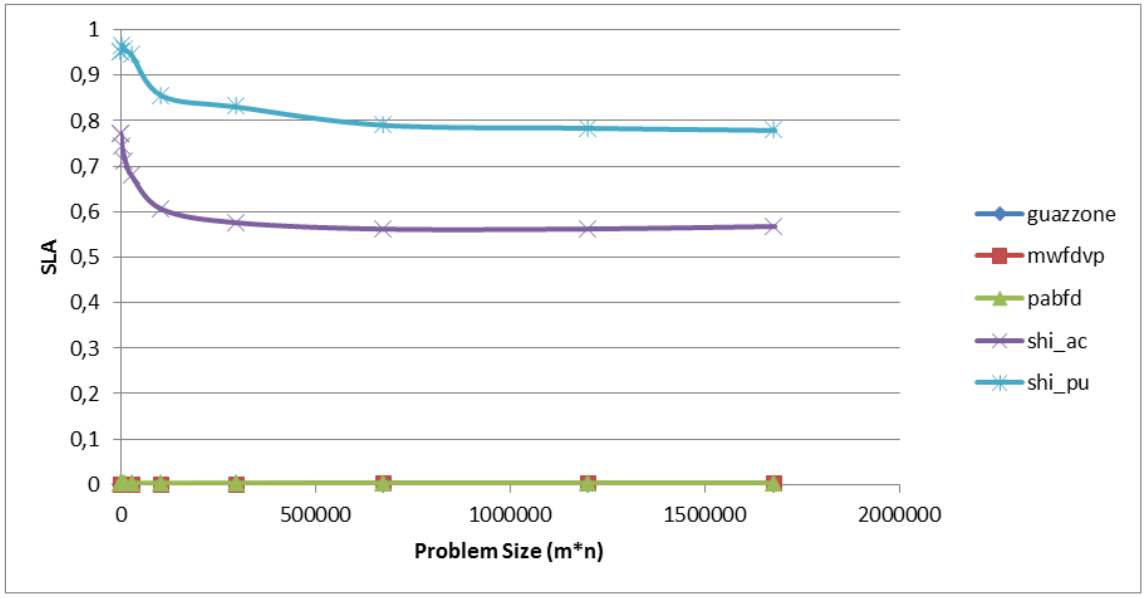

Figure 5-17: SLA in Scenario #2

5.5.5 Migrations

Figure 5-18 and Figure 5-19 show the number of migrations and the time used for these migrations respectively. In the first five test sets it decreases the number of migrations and, consequently, the migration time. However, if we consider the Figure 5-20, the similar shape of Shi approaches are clearly similar to these previous graphics, and the reason is that these algorithms spend the vast majority of migrations, while the rest have made few migrations. PABFD, MWFVP and Guazzone consider that moving VMs towards other hosts will not improve the performance due to their estimates, even though the servers could be overloaded.

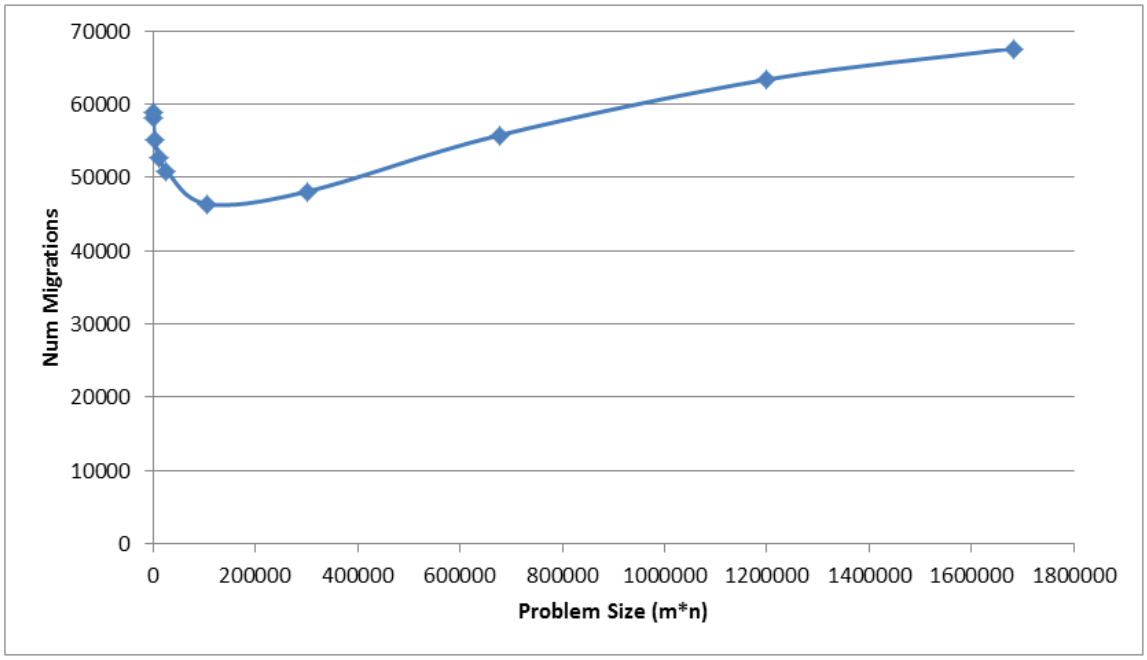

Figure 5-18: Number of migrations in Scenario #2

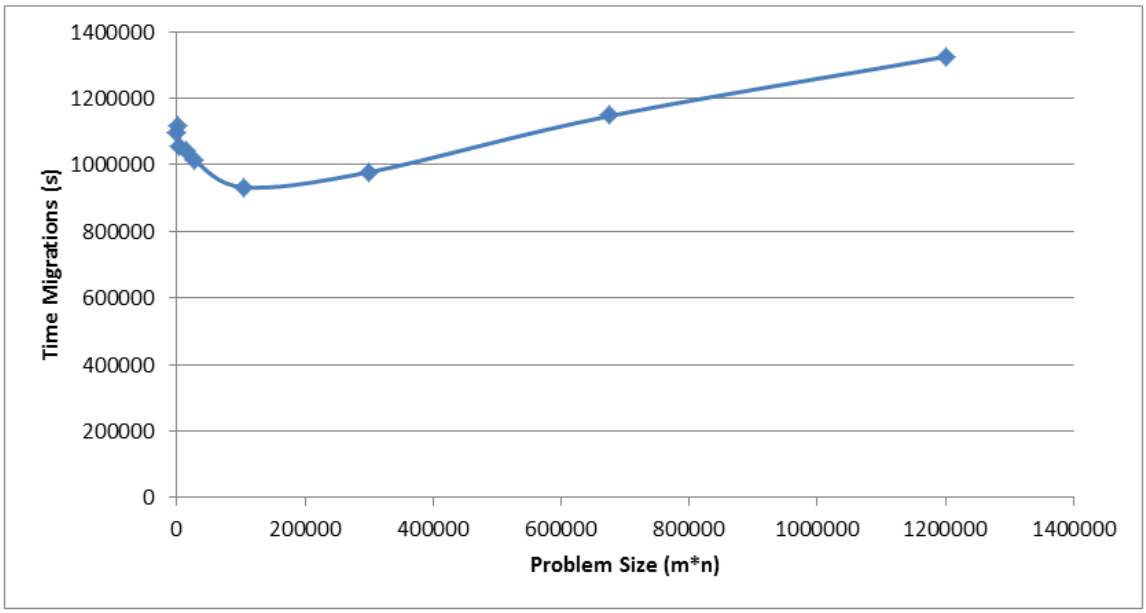

Figure 5-19: Time of migrations in Scenario #2

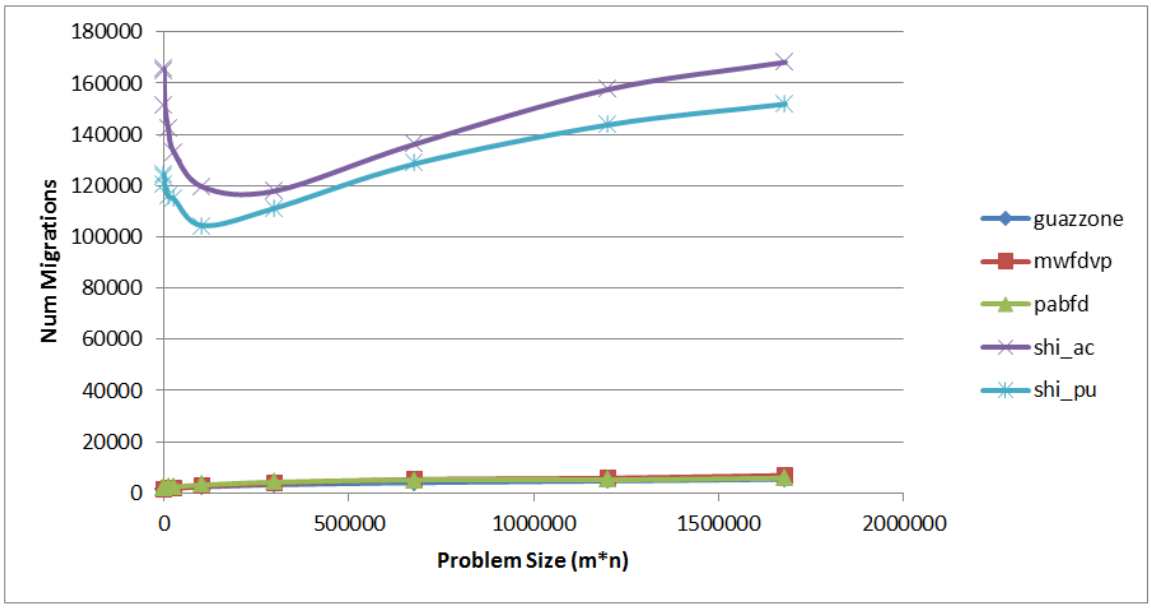

Figure 5-20: Number of Migrations per Algorithm in Scenario #2

Figure 5-21 shows that VM type 3 is the most migrated VM type. Considering the distribution of applications in 5.5.7, it seems that the migrations of VMs of this type are the VMs allocated initially in the datacenter. The reason is that these VMs have less RAM than the other VM types, and considering the VM selection policy

MMT (where the amount of memory is an important factor to select the VMs to be migrated), they are good candidates. The higher number of incoming applications balance the VMs types migrated since other factors should be considered, such as the increase of overloaded hosts, the bandwidth of the hosts are collapsed, etc.

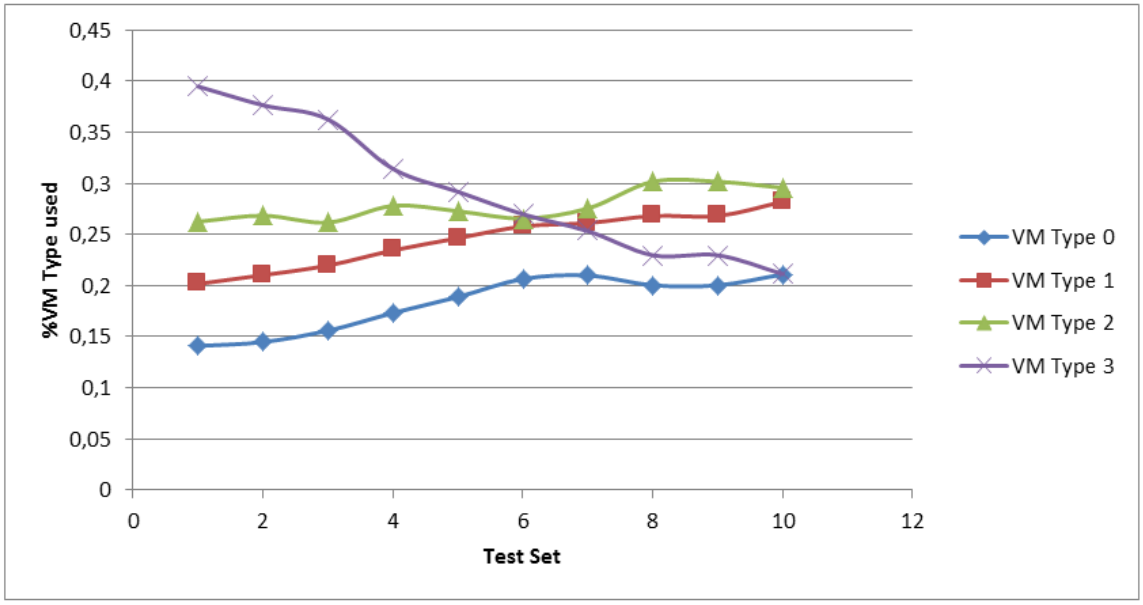

Figure 5-21: VM Types migrated in Scenario #2

5.5.6 Resource Memory Efficiency

Figure 5-22 shows that PABFD, Guazzone and MWFDVP obtain a good and stable mean resource memory efficiency in the overall test sets around the 70% of the resource used in the active hosts. Shi approaches obtain bad results since they obtain results lower than the 50% caused by the migrations that they continually do.

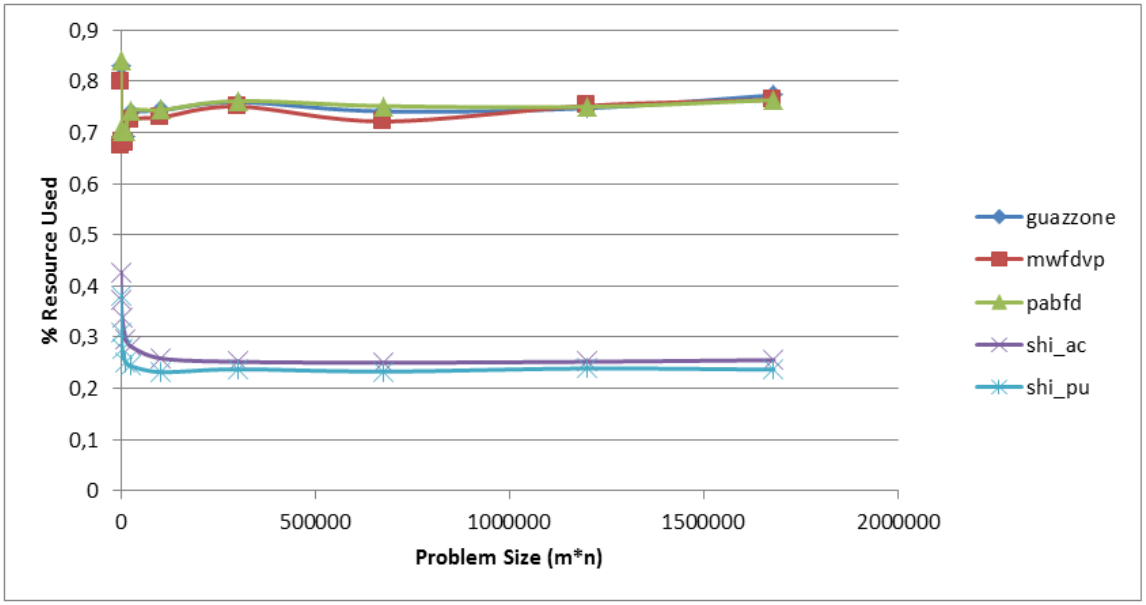

Figure 5-22: Resource memory efficiency in Scenario #2

Figure 5-23 shows that PABFD, Guazzone and MWFDVP obtain a good and stable maximum resource memory efficiency in the overall test sets around the 95% of the resource used in the active hosts. Shi approaches obtain a relevant result of 70%, but it is clearly lower than the first ones.

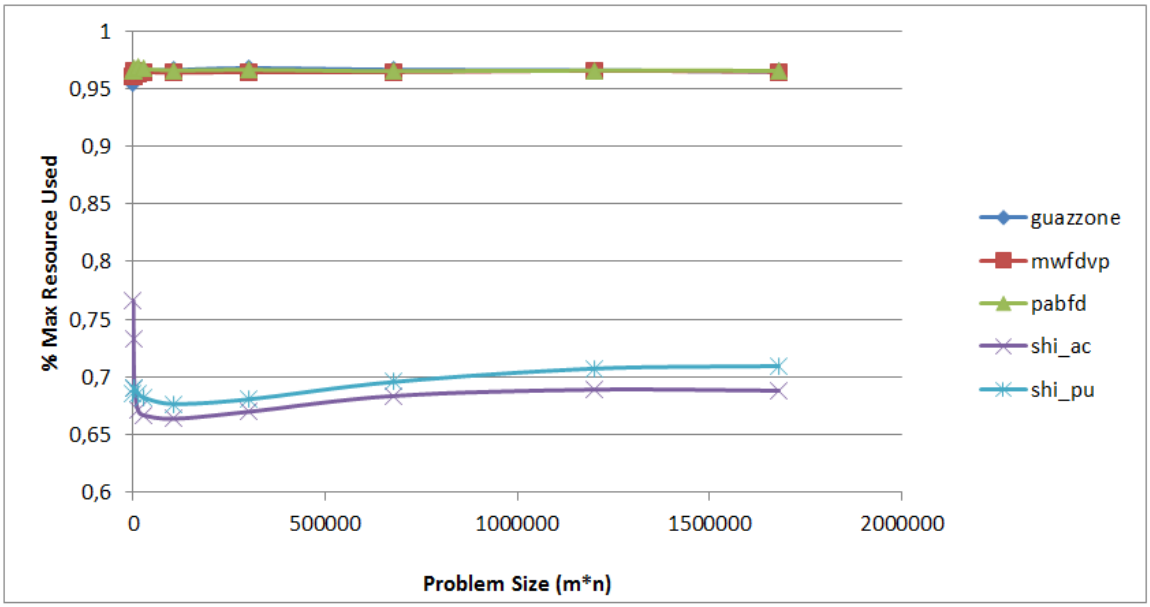

Figure 5-23: Maximum Resource Usage in Scenario #2

5.5.7 Applications Distribution on VM types

Figure 5-24 shows the distribution of applications in the different VM types considered. We can see that VM type 3 is almost ignored in the overall tests, while the vast majority of VM types selected are 1 and 2. Both VMs have high CPU capacity and memory RAM that are relevant characteristics to be considered in the application assignment process.

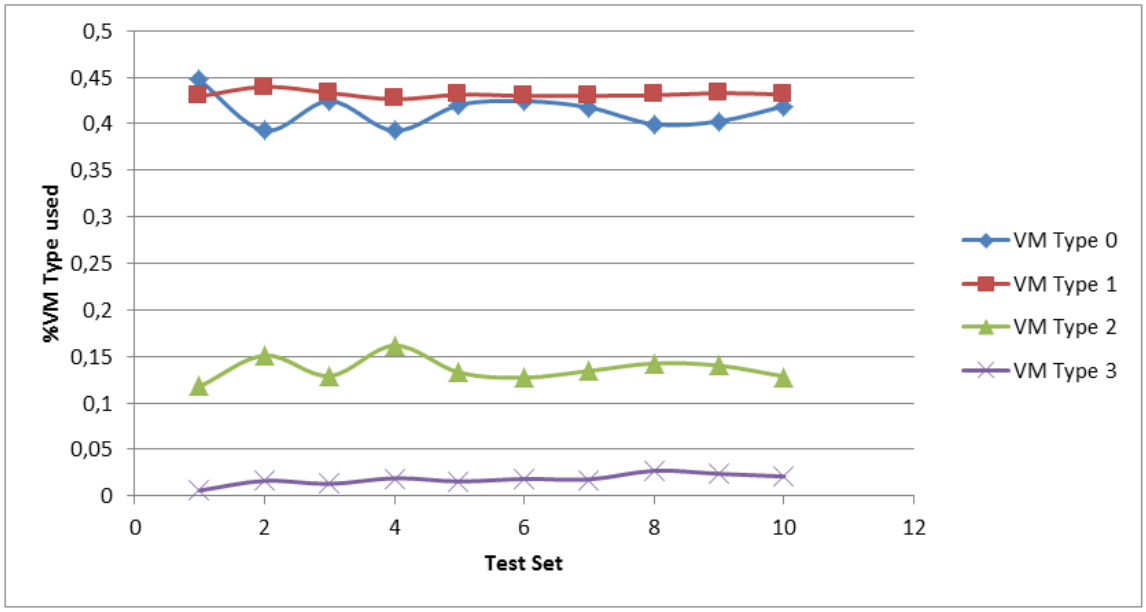

Figure 5-24: Applications Distribution in VM types in Scenario #2

5.5.8 Modeling and Prediction

5.5.8.1 Complete Dataset

Preprocessing

We checked variables with a zero variance test that results with SD_APP_MEMORY. We used Boruta and it finds that SD_APP_MEMORY and SD_APP_LENGTH are not relevant predictors as well. Finally, the statistically significant test decided the next variables to be removed: SD_APP_MEMORY, NUM_VM_MIGR_TYPE3, NUM_VM_MIGR_TYPE2 and SD_APP_LENGTH.

Exploration

The exploration of data is necessary to understand the type of data generated. A summary of the data preprocessed is shown in Figure 5-10, where some conclusions are introduced. The first one is that some relevant variables, like our response variable ENERGY_CONS, are relevantly unbalanced due to the size of the experiments produced.

Particularly, we can see that some of these values are similar from Scenario #1.

- NUM_VMS, NUM_CLOUDLETS, MEAN_APP_LENGTH, MEAN_APP_MEMORY, VM_PLACEMENT_POL are created following the known *experiment design*.

- MEAN_RESOURCE_UTILIZATION shows scattered values due the minimum, median, mean and maximum they have. The minimum and 1st quartile shows that some executions have a very bad resource utilization, while the 3rd quartile and maximum value show a good resource utilization. Both quartiles do not differ too much of the mean.

- MAX_RESOURCE_UTILIZATION shows that most instances have a good resource utilization because the majority of values are closer to 1.

- ENERGY_CONS SLA, APPS_VMTYPE0, APPS_VMTYPE1, APPS_VMTYPE2, APPS_VMTYPE3, NUM_VM_MIGR, TIME_VM_MIGR, NUM_VM_MIGR_TYPE0 and NUM_VM_MIGR_TYPE1 are unbalanced. The median is much lower than the 3rd quartile, and specially the maximum value. The main reason would be the variations of the experiment size or the algorithms applied.

```
    NUM_VMS       NUM_CLOUDLETS  MEAN_APP_LENGTH   MEAN_APP_MEMORY  APPS_VMTYPE0     APPS_VMTYPE1    APPS_VMTYPE2
Min.   :  12.0   Min.   :  15   Min.   : 400000   Min.   :250     Min.   :   0.0   Min.   :  0.0   Min.   :  0.00
1st Qu.:  52.0   1st Qu.:  60   1st Qu.:2300000   1st Qu.:250     1st Qu.:  21.0   1st Qu.: 26.0   1st Qu.:  1.00
Median : 226.0   Median : 265   Median :4200000   Median :500     Median :  93.0   Median :  99.0   Median :  9.00
Mean   : 409.2   Mean   : 486   Mean   :4200000   Mean   :500     Mean   : 199.7   Mean   :209.6   Mean   : 65.95
3rd Qu.: 752.0   3rd Qu.: 900   3rd Qu.:6100000   3rd Qu.:750     3rd Qu.: 301.0   3rd Qu.:349.0   3rd Qu.: 57.00
Max.   :1200.0   Max.   :1400   Max.   :8000000   Max.   :750     Max.   :1060.0   Max.   :913.0   Max.   :497.00
 APPS_VMTYPE3   MEAN_RESOURCE_UTILIZATION MAX_RESOURCE_UTILIZATION VM_PLACEMENT_POL SIMULATION_TIME   ENERGY_CONS
Min.   :  0.0   Min.   :0.0640            Min.   :0.4410           guazzone:1800    Min.   :  4080   Min.   : 2029
1st Qu.:  0.0   1st Qu.:0.2950            1st Qu.:0.7320           mwfdvp  :1800    1st Qu.: 62670   1st Qu.:10714
Median :  0.0   Median :0.6570            Median :0.9450           pabfd   :1800    Median :114600   Median :15328
Mean   : 10.6   Mean   :0.5554            Mean   :0.8551           shi_ac  :1800    Mean   :131665   Mean   :17901
3rd Qu.:  8.0   3rd Qu.:0.7700            3rd Qu.:0.9720           shi_pu  :1800    3rd Qu.:184230   3rd Qu.:22136
Max.   :355.0   Max.   :0.9640            Max.   :1.0000                            Max.   :549600   Max.   :65602
      SLA             NUM_VM_MIGR      TIME_VM_MIGR      NUM_VM_MIGR_TYPE0 NUM_VM_MIGR_TYPE1
Min.   :0.000008   Min.   :   780   Min.   :   14496   Min.   :    92    Min.   :   108.0
1st Qu.:0.000181   1st Qu.:  1607   1st Qu.:   30141   1st Qu.:   276    1st Qu.:   307.8
Median :0.003329   Median :  3940   Median :   65992   Median :  1222    Median :   847.0
Mean   :0.308128   Mean   : 55685   Mean   : 1112974   Mean   : 10212    Mean   : 13654.6
3rd Qu.:0.713952   3rd Qu.: 77922   3rd Qu.: 1590360   3rd Qu.: 17326    3rd Qu.: 24623.2
Max.   :1.323493   Max.   :871453   Max.   :18892054   Max.   :106580    Max.   :213312.0
```

Figure 5-25: Summary of the preprocessed dataset in Scenario #2

Due to the fact that the majority of variables are continuous except VM_PLACEMENT_POL, a correlation matrix is processed as it is shown in Figure 5-25. We make some conclusions:

- NUM_VMS, NUM_CLOUDLETS, APPS_VMTYPE0 and APPS_VMTYPE1 are highly correlated, which shows that the increment of size affects consequently to large number of assignations of VMs of the first types.

- ENERGY_CONS is quite correlated with NUM_VMS, NUM_CLOUDLETS and APPS_VMTYPE1 derived for the increase of the experiment size. For the same reasons, the migration attributes and the simulation time are slightly correlated with the ENERGY_CONS as well.

- SLA and MEAN_RESOURCE_UTILIZATION or MAX_RESOURCE_UTILIZATION are negatively correlated. The higher the resource utilization, the less SLA. It seems derived for the algorithms used in the implementation where two of them give high SLA and low resource utilization, while the others produce opposite results.

- NUM_VM_MIGR, TIME_VM_MIGR, NUM_VM_MIGR_TYPE0 and NUM_VM_MIGR_TYPE1 are highly correlated. The higher number of VMs migrated, the more time will be consumed, and the number of migrations of the types of the VMs are increased as well.

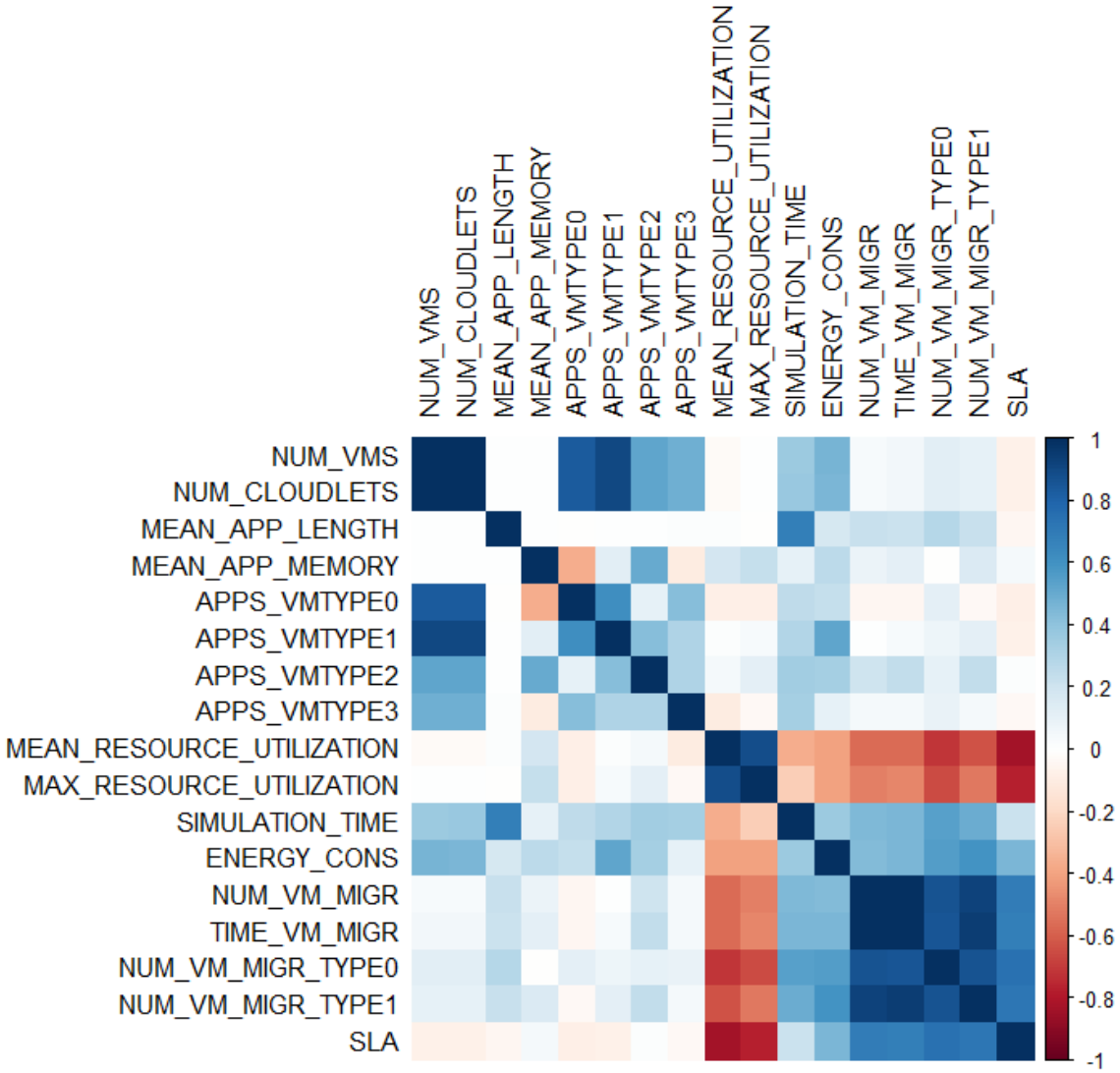

Figure 5-26: Correlation matrix of the preprocessed dataset in scenario #2

Modeling

The comparison of the quality of the prediction methods used in the complete dataset are shown in Table 5-10.

Log-linear and linear model produce quite bad results considering the mean of ENERGY_CONS, since the errors are higher than 3000 and RMSE almost arrives to 4500. The error rate is between 17% and 26%, which is too much to consider them good models.

Random Forest produces a MAE lower than 1500, and RMSE near to 2000. It seems a good improvement compared with linear methods. The error rate is between 7% and 11%, which is not a bad result considering the unbalanced values.

	RMSE	MAE	R2
Linear Model	4692.63	3366.12	0.79
Log-Linear Model	4516.73	3050.41	0.81
Random Forest	2046.30	1402.19	-

Table 5.10: Machine Learning Algorithms comparison in the complete dataset in scenario #2

5.5.8.2 Simple Dataset

Preprocessing

We checked variables with zero variance test that results with SD_APP_MEMORY. We used Boruta and it finds that SD_APP_MEMORY and SD_APP_LENGTH are not relevant predictors as well. Finally, the statistically significant test select the next variables to be removed: SD_APP_MEMORY and SDD_APP_LENGTH.

Exploration

The exploration is shown in the complete dataset, where the resulted variables in this dataset should be analyzed considering the summary statistics and correlation matrix already obtained in the previous dataset.

Modeling

The comparison of the quality of the prediction methods used in the simple dataset are shown in Table 5-11. Linear methods produce very bad results considering the mean of ENERGY_CONS, which could produce an error rate of 23%-35%.

Random Forest produces a MAE near to 2200, and the RMSE is higher than 3000. The error rate is between 12% and 16%, which it does not seem a good model.

	RMSE	MAE	R2
Linear Model	6282.79	4701.46	0.64
Log-Linear Model	5780.33	4125.83	0.70
Random Forest	3155.33	2201.35	-

Table 5.11: Machine Learning Algorithms comparison in the simple dataset in scenario #2

5.5.8.3 Conclusions

Complete datasets obtain better results than simple datasets because more information is considered in the modeling. The best model is the Random Forest using the complete dataset that shows an error rate around 8%. This error should not be ignored, but it gives a near estimation of the energy consumption that could be useful. The linear methods show large errors that should not be considered as good models to predict accurately our data.

Chapter 6

Conclusions

CloudSim is a powerful framework for simulating an Infrastructure as a Service (IaaS) service cloud to extract relevant information of the behaviour in a datacenter. We selected some literature algorithms to deal with some complexity problems and they were tested in practice.

Within this project, we studied deeply how CloudSim works and what parts of its internal process can be modified and extended to achieve our goals. Some algorithms were discussed in detail, showing which of them are associated to the determined optimization process. The main objectives, as presented in *1.2.*, can be considered accomplished.

Two scenarios were considered for conducting our experiments, and we calibrated our expectations after the results. The first scenario was an empty datacenter which received the amount of applications to process in a fixed submission time. The second scenario used the same configuration but it added an initial loading in some hosts of the datacenter. The test cases considered different sizes that were formed by the number of applications and VMs executed.

From the results evaluation, we come to the conclusion that Guazzone algorithm [37], which deals with the VM placement problem, is the most interesting approach in terms of energy consumption, solution time, quality of service and resource memory efficiency based in our tested scenarios. However, PABFD approach from Beloglazov et al. [22] (authors of power package CloudSim), and Chowdhury [36] methods almost

obtained similar results. Shi approaches presented in [38], and discussed in [26] as one of the best candidates for energy-aware consumption, did not obtain good results in our experiments. Shi obtained a large number of migrations producing overheads that aggravate metrics such as energy consumption or quality of service. Algorithms should be studied and considered in different specific scenarios since maybe they will not suit in the datacenter infrastructure contemplated.

We considered the Repairing Genetic Algorithm [1] to deal with the application assignment problem. It maximized the distribution of applications to the VMs types with higher CPUs capacity while maintaining the resource memory utilization. The scalability of the experiment showed a linear increment when an intermediate test set was executed. However, it grew exponentially in the first test sets due to the complexity of the applications against the modest infrastructure used in the experiments.

Some models were considered to predict the energy consumption based on the simulation process. The non-linear method showed better results than the linear ones. Two datasets were examined: one with the overall variables produced in the execution, and a simpler one which only considered the input variables of the experiment. The first dataset obtained better predictions showing the necessity of the learned variables of the system. However, the best model produced an error rate around 8% that we should not ignore. The prediction was difficult because of the unbalanced values that the dataset contained.

Research Area

In our experiments, some predefined algorithms were selected to study different criteria. However, other algorithms from the literature that focus on a determined criteria could be considered to analyze and extend this research. The scenarios were also predefined, and others could be useful to analyze different possible situations, such as a higher rate of incoming applications or changing the VMs types. The models were built with the selection of some variables calculated and obtained from the simulator. Due the large amount of variables in the simulator, we could not select all of them. However, others variables could be joined to analyze whether they could

improve the prediction.

Checking the latest version of CloudSim, the Container as a Service modules were implemented. Another study could be the analysis of these functionalities in this service model against the Infrastructure as a Service model.

Chapter 7

Economic Report

In this chapter, the main developing tasks identified for this project and their distribution along time are explained in the first section, while in the second the costs of development and infrastructure are detailed.

7.1 Project planning

In order to achieve the objectives mentioned in section *1.2.*, the main tasks were identified, and some estimations were given for each of them in order to evaluate the necessary time for the development of the project. Estimations were expressed in number of days to complete them. Even though some difficulties and unplanned work should be expected, giving a time estimation for each task seemed more relevant in order to be able to evaluate the progress. The project was started in May 2017, when the classes were almost over to focus in the project. This project can be divided in different steps shown below. The first step of the project is the analysis of the main elements involved. A datacenter simulator is selected from large possibilities, and how to face the complexities of the project is contemplated, from the framework to use until their corresponding libraries.

Once the basic steps are decided, the next step is the familiarization of the selected framework CloudSim following a learning process, where the main classes and packages should be analyzed in detail. In this process, a large document is considered

to understand how CloudSim works and what properties and characteristics define its power purposes. A research for the algorithms to be implemented is also considered in this step, recognizing the main advantages and disadvantages of each method.

The next step is the implementation of those relevant algorithms to be considered in our study. To achieve that, it is necessary to follow the indications that have been learned in the previous step, and follow their corresponding steps in the literature. In this step is also considered the generation of the instances using the infrastructure of RDLab, which needs a cluster familiarization of usage.

The last step is conducting the experiments and evaluate the performance of overall algorithms implemented in CloudSim. Besides, a model to predict the energy consumption based in their corresponding scenario and algorithms is implemented.

The writing of this thesis begins before implementing and conducting all the experiments, showing the steps done and allowing to do it in parallel for the large quantity of time required.

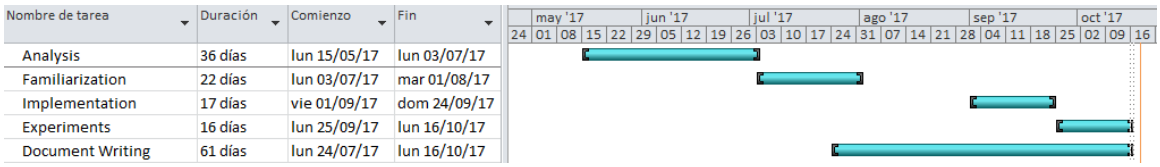

Figure 7-1: Project Planning

7.2 Costs

The costs of this project are divided between development and infrastructural costs.

Development costs

The tasks are mainly engineering tasks performed by a Software Engineer. According to Payscale platform [44], an Entry-Level Software Engineer working in Barcelona is around 2500€ per month. Considering a four months activity on a daily working bases, the development cost would be as follows:

Salary per month	Number of Months	Total
2500€	4	10.000€

Table 7.1: Cost of the project.

Infrastructural costs

The infrastructural costs are divided between the software and the hardware costs. The software used in the project is freeware, so the costs are zero. Software includes tools such as CloudSim, RStudio and Eclipse.

The hardware costs of this project were also zero since the laptop and the RDLab infrastructure used were provided freely. Fortunately, there were no monetary costs for these purposes thanks for the kindness of the research of the university.

Finally, the total development cost will be around the 10.000€.

Glossary

API Application Programming Interface.

CloudSim Cloud Simulation Toolkit for Modelling and Simulation of Clouds.

IaaS Infrastructure as a Service.

IoT Internet of Things.

IT Information Technology.

JAR Java Archive.

MIPS Million Instructions per Second.

NIST National Institute of Standards and Technologies.

OpEx Operating expense.

PaaS Platform as a Service.

PE Process Element.

PM Power Management.

QoS Quality of Service.

RAM Random Access Memory.

SaaS Software as a Service.

SLA Service Level Agreement.

VM Virtual machine.

Bibliography

[1] Meera Vasudevan. *Profile-based application management for green data centres*. PhD thesis, Queensland University of Technology, 2016.

[2] Imran Ghani, Naghmeh Niknejad, and Ryul Seung. Energy saving in green cloud computing data centers: A review. In *Journal of Theoretical and Applied Information Technology*, volume 1074. 05 2015.

[3] Shaden M. AlIsmail and Heba A. Kurdi. Review of energy reduction techniques for green cloud computing. In *International Journal of Advanced Computer Science and Applications(ijacsa)*, volume 7. 2016.

[4] Shalabh Agarwal, Arnab Datta, and Asoke Nath. Impact of green computing in it industry to make eco friendly environment. In *Journal of Global Research in Computer Science*, volume 5, pages 5–10. 05 2014.

[5] Department of Energy Announcement from Secretary Chu. [Online]. `https://energy.gov/articles/secretary-chu-announces-47-million-improve-efficiency-information-technology-and`, October 2010.

[6] Murat Caliskan, Mustafa Ozsiginan, and Emin Kugu. Benefits of the virtualization technologies with intrusion detection and prevention systems. In *2013 7th International Conference on Application of Information and Communication Technologies*, pages 1–5, Oct 2013.

[7] Peter Mell and Tim Grance. The nist definition of cloud computing. In *Communications of the ACM*, volume 53. 01 2011.

[8] Grid Computing and Distributed Computing. [Online]. `http://www.maxi-pedia.com/Grid+computing+distributed+computing`, October 2017.

[9] Whatiscloud. [Online]. `http://whatiscloud.com/cloud_deployment_models/index`, October 2017.

[10] Georgia Sakellari and George Loukas. A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing. In *Simulation Modelling Practice and Theory*, volume 39, page 92–103. 12 2013.

[11] Simjava. [Online]. `http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava`, October 2017.

[12] CloudSim. [Online]. http://www.cloudbus.org/, October 2017.

[13] GreenCloud. [Online]. https://greencloud.gforge.uni.lu, October 2017.

[14] The Network Simulator. [Online]. https://www.isi.edu/nsnam/ns/, October 2017.

[15] Icancloud. [Online]. https://www.arcos.inf.uc3m.es/old/icancloud/Home.html, October 2017.

[16] Lovejit Jhajj and Sarbjeet Singh. A survey of workflow scheduling algorithms and research issues. In *International Journal of Computer Applications*, volume 74. 07 2013.

[17] Mustafizur Rahman, Rafiul Hassan, Rajiv Ranjan, and Rajkumar Buyya. Adaptive workflow scheduling for dynamic grid and cloud computing environment. *Concurrency and Computation: Practice and Experience*, 25(13):1816–1842, 2013.

[18] Anju Bala and Dr.Inderveer Chana. A survey of various workflow scheduling algorithms in cloud environment. *IJCA Proceedings on 2nd National Conference on Information and Communication Technology*, NCICT(4):26–30, November 2011.

[19] Vijindra and Sudhir Shenai. Survey on scheduling issues in cloud computing. *Procedia Engineering*, 38(Supplement C):2881 – 2888, 2012.

[20] Jatin Jain. A hybrid approach for time efficient and reliable workflow scheduling in cloud computing. Master's thesis, Thapar University, 2014.

[21] Planetlab. [Online]. https://www.planet-lab.org/, October 2017.

[22] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13), 2011.

[23] KyoungSoo Park and Vivek S. Pai. Comon: a mostly-scalable monitoring system for planetlab. *Operating Systems Review*, 40(1):65–74, 2006.

[24] Bitbrains. [Online]. http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains, October 2017.

[25] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *ACM Symposium on Cloud Computing, SOCC '12, San Jose, CA, USA, October 14-17, 2012*, page 7, 2012.

[26] Zoltán Ádám Mann and Máté Szabó. Which is the best algorithm for virtual machine placement optimization? *Concurrency and Computation: Practice and Experience*, 29(10), 2017.

[27] Google Cluster Data Set. [Online]. `https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md`, October 2017.

[28] Rodrigo N. Calheiros, R Ranjan, Anton Beloglazov, Cesar De Rose, and Rajkumar Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. In *Software Practice and Experience*, volume 41, pages 23–50. 01 2011.

[29] Preemptive and Non preemptive Tasks. [Online]. `http://techdifferences.com/difference-between-preemptive-and-non-preemptive-scheduling-in-os.html`, October 2017.

[30] TutorialsCloudSim. [Online]. `http://www.superwits.com/library/cloudsim-simulation-framework`, October 2017.

[31] Khalid Jebari. Selection methods for genetic algorithms. In *International Journal of Emerging Sciences*, volume 3, pages 333–344. 12 2013.

[32] William S. Cleveland. Robust locally weighted regression and smoothing scatterplots. In *Journal of the American Statistical Association*, volume 74, pages 829–836. 12 1979.

[33] Anton Beloglazov. *Energy-efficient management of virtual machines in data centers for cloud computing*. PhD thesis, University of Melbourne, Australia, 2013.

[34] Bin-packing problem. `http://www.or.deis.unibo.it/kp/Chapter8.pdf`, October 2017.

[35] Liang Luo, Wenjun Wu, W.T. Tsai, Dichen Di, and Fei Zhang. Simulation of power consumption of cloud data centers. *Simulation Modelling Practice and Theory*, 39(Supplement C):152 – 171, 2013. S.I.Energy efficiency in grids and clouds.

[36] Mohammed Rashid Chowdhury, Mohammad Raihan Mahmud, and Rashedur M. Rahman. Study and performance analysis of various VM placement strategies. In *16th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2015, Takamatsu, Japan, June 1-3, 2015*, pages 411–416, 2015.

[37] Marco Guazzone, Cosimo Anglano, and Massimo Canonico. Exploiting VM migration for the automated power and performance management of green cloud computing systems. In *Energy Efficient Data Centers - First International Workshop, E2DC 2012, Madrid, Spain, Mai 8, 2012, Revised Selected Papers*, pages 81–92, 2012.

[38] Lei Shi, John Furlong, and Runxin Wang. Empirical evaluation of vector bin packing algorithms for energy efficient data centers. In *2013 IEEE Symposium on Computers and Communications, ISCC 2013, Split, Croatia, 7-10 July, 2013*, pages 9–15, 2013.

[39] Rstudio. [Online]. `https://www.rstudio.com/`, October 2017.

[40] Rdlab. [Online]. `https://rdlab.cs.upc.edu/index.php/en`, October 2017.

[41] Amazon EC2 Instances Types. [Online]. `https://aws.amazon.com/es/ec2/instance-types`, October 2017.

[42] Miron Bartosz Kursa and Witold Remigiusz Rudnicki. Boruta package. `https://cran.r-project.org/web/packages/Boruta/Boruta.pdf`, October 2017.

[43] Richard L. Lieber. Statistical significance and statistical power in hypothesis testing. *Journal of Orthopaedic Research*, 8(2):304–309, 3 1990.

[44] Payscale. [Online]. `https://www.payscale.com/`, October 2017.